Σύντομος οδηγός εκμάθησης για αρχάριους

Η PYTHON ΜΕ ΑΠΛΑ ΛΟΓΙΑ



print()

int(input())

while

range()

Αλέξανδρος Κοφτερός, PhD

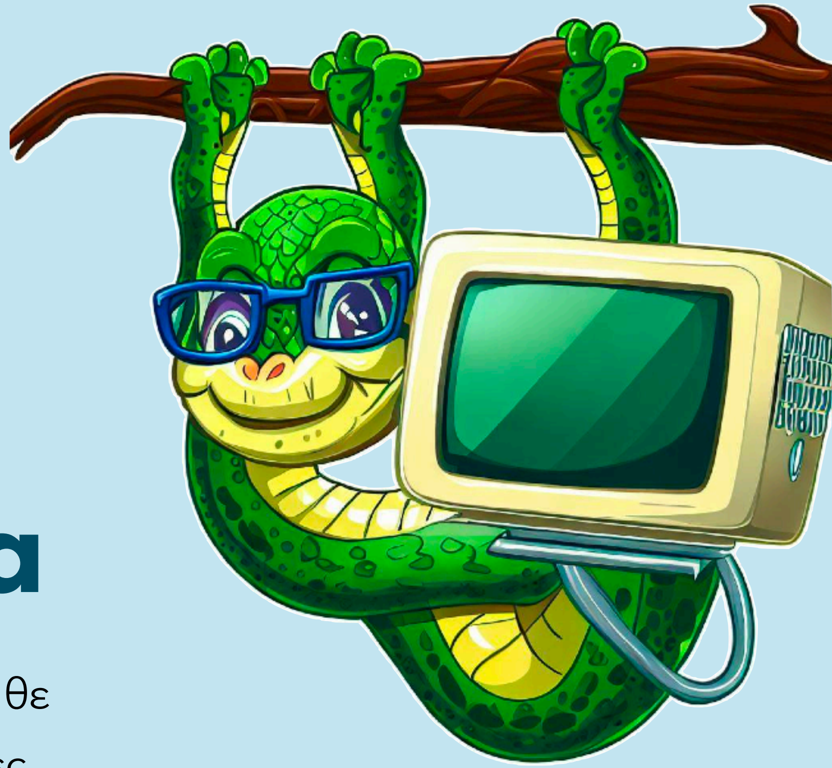
Η Python με απλά λόγια

Σύντομος οδηγός εκμάθησης
για αρχάριους

Nicosia, 2023
978-9925-8055-0-1

All images created with Microsoft Bing Create.





Ευχαριστώ ιδιαίτερα

για τον πολύτιμο χρόνο που αφιέρωσαν να διαβάζουν κάθε έκδοση του οδηγού και να στέλνουν συνεχώς αναλυτικές εισηγήσεις και σχόλια:

Πόλα Μισθού, εκπαιδευτικός (Ελλάδα)

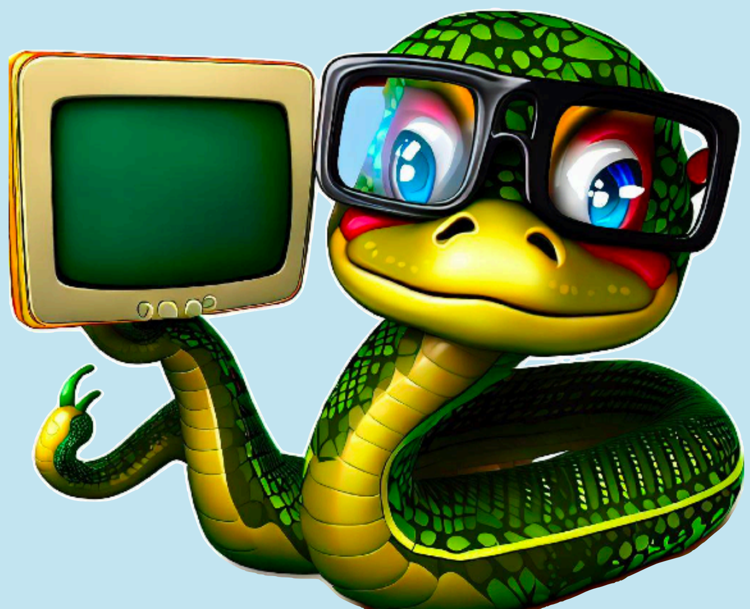
Βάσω Σέρβου, εκπαιδευτικός (Ελλάδα)

Διαμάντω Γεωργίου, εκπαιδευτικός (Κύπρος)

Αντώνη Φοινικαρίδη, IT (Κύπρος)

Γιώργο Κυπριανού, εκπαιδευτικός (Κύπρος)

Μάριο Χαραλάμπους, γραφίστας (Κύπρος)





- Στο βιβλίο αυτό θα γνωρίσουμε βασικά χαρακτηριστικά των γλωσσών προγραμματισμού
- Θα γνωρίσουμε την ιστορία της Python
- Θα μάθουμε και θα χρησιμοποιήσουμε εντολές της Python
- Θα δημιουργήσουμε γεωμετρικά σχήματα με την Python
- Θα προγραμματίσουμε συσκευές όπως micro:bit και MeetEdison με εντολές σε Python

Σχετικά με το βιβλίο

Καλωσορίσατε στο βιβλίο μας για την Python! Πρόκειται για την πρώτη προσπάθεια δημιουργίας ενός απλού οδηγού εισαγωγής στη συγκεκριμένη γλώσσα προγραμματισμού, που απευθύνεται σε εκπαιδευτικούς αλλά και μαθητές και μαθήτριες 10+ ετών.

Το βιβλίο αυτό απευθύνεται κυρίως σε αρχάριους σε θέματα προγραμματισμού, που ξεκινούν τα πρώτα τους βήματα και επιθυμούν να γνωρίσουν μια αρκετά ικανή γλώσσα.

Το βιβλίο είναι χωρισμένο σε τέσσερα μέρη: στο πρώτο μέρος θα γνωρίσουμε περισσότερες πληροφορίες για τις γλώσσες προγραμματισμού, με έμφαση στην Python. Στο δεύτερο μέρος θα γνωρίσουμε βασικές εντολές της Python, μέσα από απλά παραδείγματα. Στο τρίτο μέρος θα αναφερθούμε σε πρόσθετες λειτουργίες της Python, με έμφαση στη δημιουργία γεωμετρικών σχημάτων.

Στο τέταρτο μέρος θα γνωρίσουμε περιβάλλοντα προγραμματισμού σε Python για εφαρμογές IoT και εκπαιδευτικής ρομποτικής (BBC Micro:bit, MeetEdison).

Καλωσορίσατε!



Ο οδηγός (ή το βιβλίο, αν θέλετε) που έχετε μπροστά σας, ξεκίνησε τον Ιούλιο του 2021 ως βοήθημα (σημειώσεις) για να μάθει Python ο γιος μου. Στην πορεία, ο Κωνσταντίνος ασχολήθηκε με C++, που του αρέσει ιδιαίτερα, ενώ εγώ έβαλα στον "πάγο" το βιβλίο της Python για να επικεντρωθώ στο εκπαιδευτικό πρόγραμμα του Μουσείου Υπολογιστών. Δύο χρόνια μετά, και αφού πέρασε από ΠΑΡΑ πολλές αλλαγές, είναι έτοιμη η πρώτη έκδοση του βιβλίου μου.

Η λογική της δημιουργίας του είναι απλή: ήθελα να επιστρέψουμε στη λογική της δεκαετίας του 1980, όπου ανοίγαμε τον υπολογιστή μας (Spectrum, AMSTRAD CPC, Commodore 64 κ.α.) και βλέπαμε μπροστά μας τις επιλογές της BASIC. Για κάποιο λόγο, μας άρεσε όλους να δημιουργούμε έναν ατέρμονο βρόχο (infinite loop).

Με την ίδια λογική, έγραψα ένα βιβλίο που απευθύνεται σε μικρές ηλικίες (10+) και επιχειρεί να βάλει τα δάκτυλά μας ξανά πάνω στο πληκτρολόγιο!

Python! Monty Python!





ΜΕΡΟΣ Α΄: ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΥΠΟΛΟΓΙΣΤΩΝ

1. Εισαγωγή στον Προγραμματισμό

```
10 |  print ("And what is the use of a book
20 |  without pictures or conversation?")
50 | #Alice in Wonderland
```

1. Τι είναι η Python

Όλοι γνωρίζουν ότι οι πύθωνες είναι συμπαθητικά ζώα. Βέβαια, δε θα σας συμβουλεύαμε ποτέ να πλησιάσετε έναν, ούτε για αστείο. Και αν κάποτε βρεθείτε σε χώρα που έχει πύθωνες, μάλλον να αποφύγετε να πλησιάσετε σε μέρη που ζουν. Δεν τους αρέσει να μπαίνουν άγνωστοι, ειδικά άνθρωποι, στα μέρη τους!

Σε αυτό το βιβλίο, βέβαια, δε θα ασχοληθούμε με τα συμπαθητικά αυτά ζώα (τα οποία, όπως είπαμε, θα αφήσουμε στην ησυχία τους), αλλά με μια **γλώσσα προγραμματισμού**, ιδιαίτερα διαδεδομένη και ικανή για να δημιουργήσουμε πολύπλοκες εφαρμογές.

Το όνομα της το πήρε, όχι από το συμπαθέστατο αυτό ζώο, αλλά από τους θρυλικούς Monty Python! Ω, ναι, αυτό είναι το "μυστικό" πίσω από το όνομα της γλώσσας αυτής!



Τι θα γνωρίσουμε:

Στο **Κεφάλαιο 1: "Εισαγωγή στον Προγραμματισμό"**, θα γνωρίσουμε:

- Τι είναι οι γλώσσες προγραμματισμού
- Πώς επικοινωνεί ο άνθρωπος με τον υπολογιστή μέσω του προγραμματισμού
- Περιβάλλοντα προγραμματισμού που χρησιμοποιούμε στην εκπαίδευση
- Τη δημιουργία της γλώσσας Python
- Πώς να κατεβάσουμε και να εγκαταστήσουμε την Python στον υπολογιστή μας

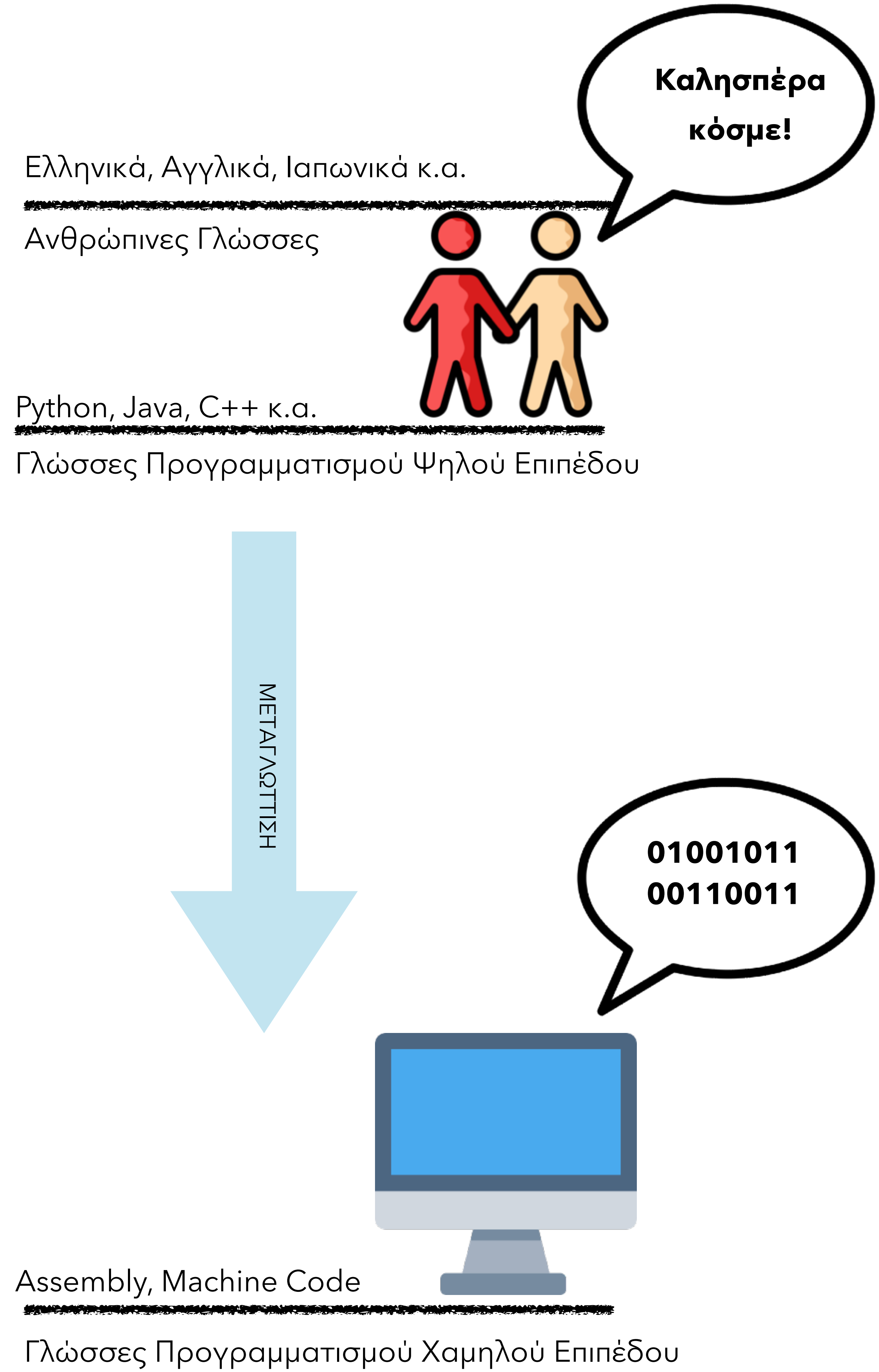


Ένα πρόγραμμα (ή κώδικας) είναι μια σειρά από εντολές που δίνουμε, συνήθως μέσω πληκτρολογίου, ώστε να εκτελεστεί μία ή περισσότερες οδηγίες.

Οι πρώτοι υπολογιστές ήταν κατασκευασμένοι να κάνουν πολύ συγκεκριμένες εργασίες (π.χ. εξειδικευμένους μαθηματικούς υπολογισμούς). Οι υπολογιστές αυτοί χρησιμοποιήθηκαν κυρίως τις δεκαετίες του 1940 και 1950 και προγραμματίζονταν σε “γλώσσα μηχανής”, μια γλώσσα που για τον υπολογιστή έχει νόημα, αλλά για τον άνθρωπο όχι (με εξαίρεση τους γνώστες της “γλώσσας μηχανής”).

Ο προγραμματισμός με αυτό τον τρόπο ήταν πολύ πολύπλοκος, και καθώς εξελίσσονταν οι υπολογιστές -και αποκτούσαν νέες δυνατότητες- ήταν πολύ δύσκολος ακόμη και για τους πλέον ικανούς προγραμματιστές.

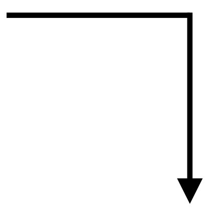
Τη δεκαετία του 1950, αναπτύχθηκαν οι πρώτες γλώσσες προγραμματισμού που ήταν κάπως απλούστερες από τη γλώσσα μηχανή (assemblers), αλλά εξακολουθούσαν να ήταν και να είναι πολύπλοκες. Την ίδια περίοδο, άρχισαν να αναπτύσσονται και γλώσσες προγραμματισμού πιο κοντά στη δική μας γλώσσα. Μερικές από τις πιο γνωστές της δεκαετίας του 1950 ήταν η FORTRAN, που αναπτύχθηκε από την IBM, και η COBOL, που αναπτύχθηκε από την Grace Hopper.



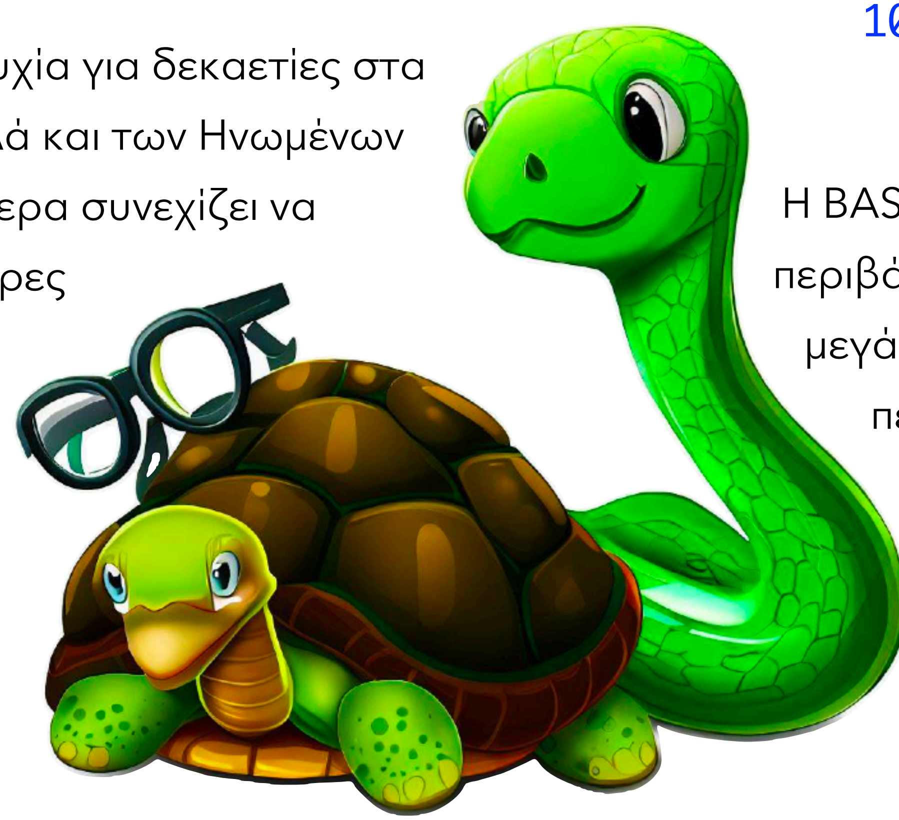
Στο βιβλίο αυτό θα ασχοληθούμε αποκλειστικά με την Python. Όμως, ο προγραμματισμός υπολογιστών στα σχολεία ξεκίνησε από τη δεκαετία του 1960 με τη δημιουργία της LOGO το 1967από τον Seymour Papert, έναν ακούραστο δάσκαλο που αγαπούσε ιδιαίτερα τα παιδιά και την εκπαίδευση.

Με τη LOGO, την οποία αρχικά προγραμμάτιζαν με ένα ρομπότ εδάφους, μπορούσαν να δημιουργήσουν γεωμετρικά σχήματα. Για παράδειγμα, για δημιουργία μιας γωνίας 90 μοιρών, έπρεπε να δοθεί η εντολή:

```
FORWARD 10
RIGHT 90
FORWARD 10
```



Η LOGO είχε μεγάλη επιτυχία για δεκαετίες στα σχολεία της Ευρώπης αλλά και των Ηνωμένων Πολιτειών. Ακόμη και σήμερα συνεχίζει να χρησιμοποιείται σε διάφορες μορφές της, συνήθως σε συνδυασμό με εντολές τύπου μπλοκ.



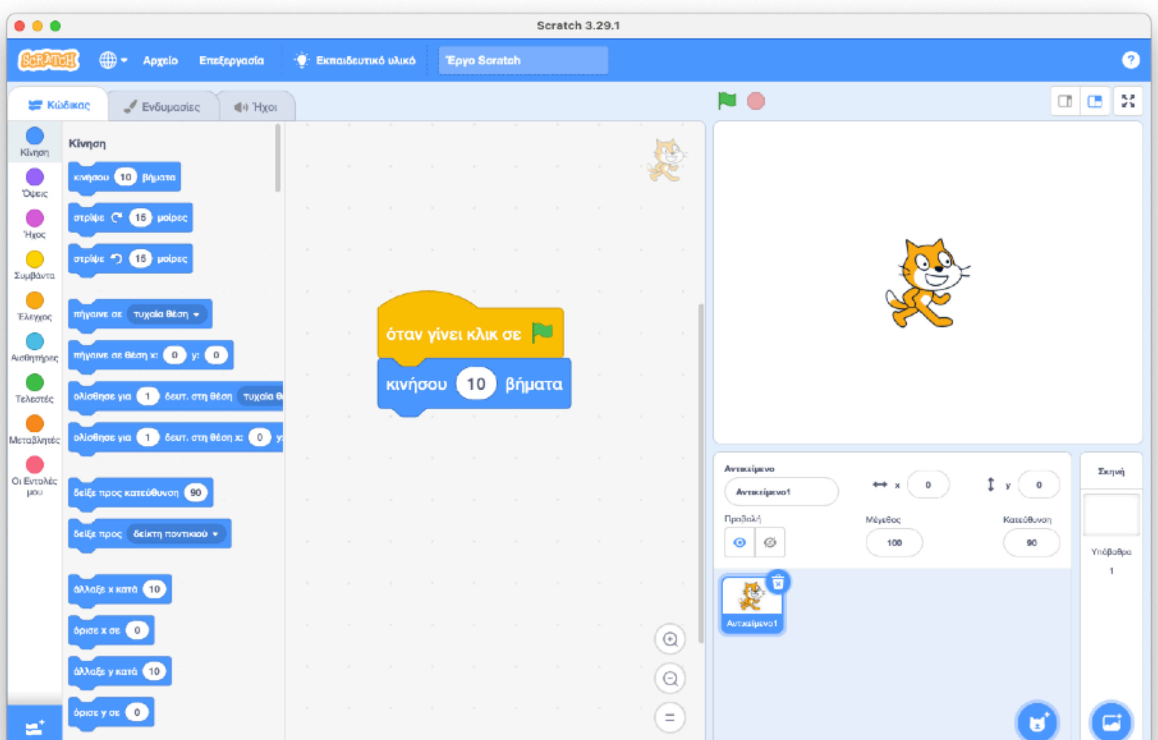
Μια άλλη γλώσσα που χρησιμοποιήθηκε αρκετά σε σχολεία, κυρίως του Ηνωμένου Βασιλείου, ήταν η BASIC (Beginners All-purpose Symbolic Instruction Code). Πρόκειται για μια από τις απλούστερες γλώσσες προγραμματισμού που δημιουργήθηκαν, και χρησιμοποιήθηκε κυρίως από αρχάριους χρήστες. Στη δεκαετία του 1980, όλοι οι προσωπικοί υπολογιστές είχαν ενσωματωμένη ή περιλάμβαναν σε δισκέτα (ή και κασέτα) μια μορφή της BASIC.

Σε BASIC, για να εμφανιστεί στην οθόνη το μήνυμα Hello World, έπρεπε να πληκτρολογήσουμε την πιο κάτω εντολή:

```
10 PRINT “HELLO WORLD”
```

Η BASIC έχει πλέον αντικατασταθεί με άλλες γλώσσες ή περιβάλλοντα προγραμματισμού που προσφέρουν μεγάλη ευκολία στον αρχάριο χρήστη, αλλά και περισσότερες δυνατότητες. Παρόλα αυτά, κάποια περιβάλλοντα ανάπτυξης λογισμικού, όπως το AOZ Studio, βασίζονται σε μια μορφή της BASIC.





Ο προγραμματισμός σε LOGO και BASIC στα σχολεία, και ειδικά σε μικρές ηλικίες, συναντούσε ένα σοβαρό (υπήρχαν και άλλα, όμως ας μείνουμε σε αυτό…) πρόβλημα: τα παιδιά έπρεπε να απομνημονεύσουν αρκετές εντολές. Και να ξέρουν όλα την αγγλική γλώσσα, καθώς δεν υπήρχε η LOGO ή η BASIC (οι εντολές τους) σε άλλες γλώσσες.

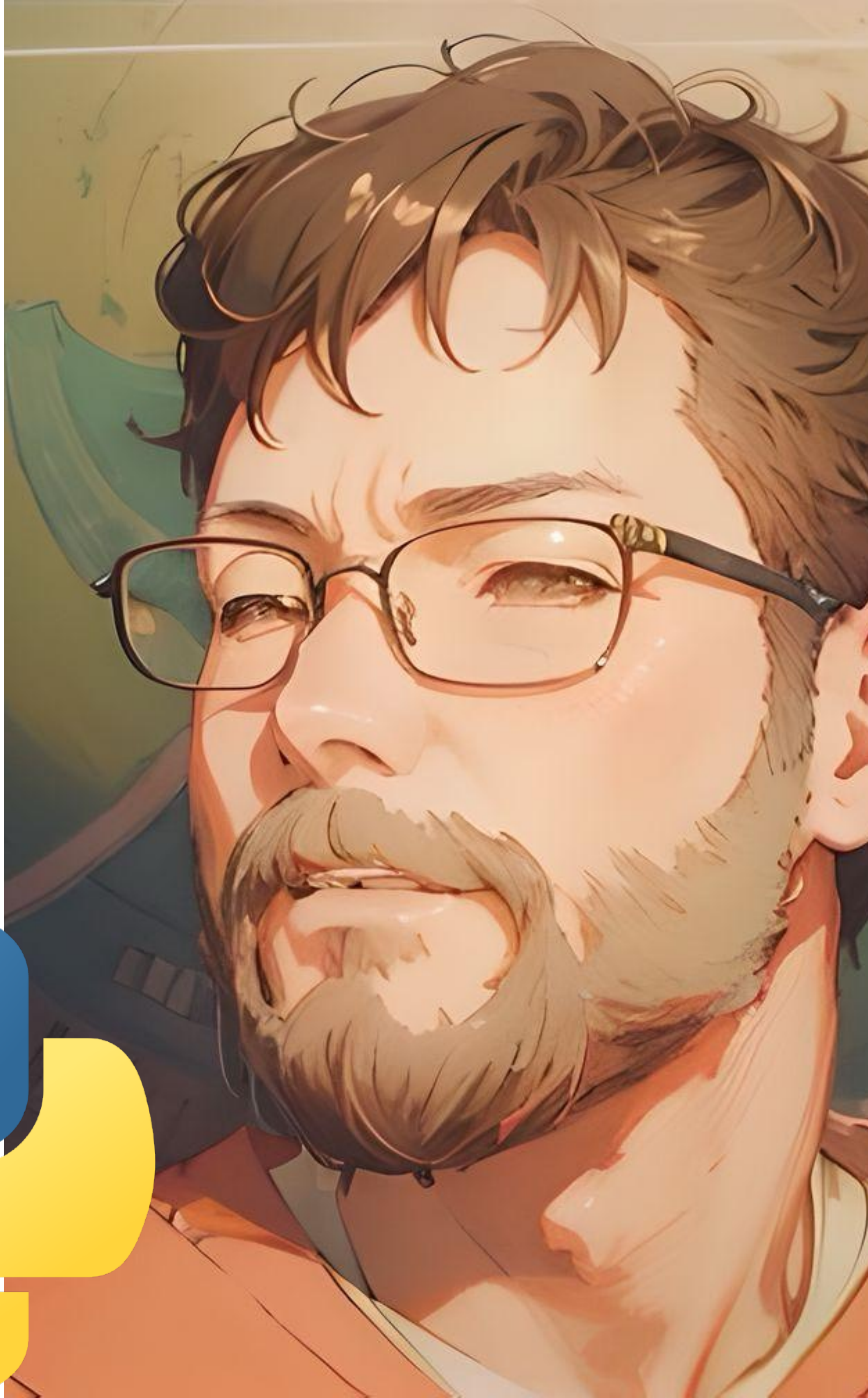
Και τα δύο αυτά προβλήματα, τουλάχιστο όσον αφορά τις μικρές τάξεις (Προδημοτική - Δ' Δημοτικού) λύθηκαν πριν το τέλος της δεκαετίας του 2000, με την ανάπτυξη του περιβάλλοντος προγραμματισμού Scratch, το οποίο βασιζόταν σε μπλοκ τα οποία μπορούμε να συνδέσουμε και να δημιουργήσουμε το πρόγραμμα μας. Ακολούθησε το Scratch Jr για tablets λίγο αργότερα, με ένα ακόμη πιο απλό περιβάλλον, κάτι που επιτρέπει εύκολα στα παιδιά της Προδημοτικής και της Α' Δημοτικού να μάθουν προγραμματισμό.

Πλέον το Scratch χρησιμοποιείται σε πολλές άλλες τεχνολογίες, όπως τα λογισμικά προγραμματισμού εκπαιδευτικών ρομπότ (MeetEdison, mBot, LEGO Spike Prime, Arduino κ.α.).

Δημιουργία Python

Η Python δημιουργήθηκε προς τα τέλη του 1980 από τον Guido van Rossum στην Ολλανδία, με την αρχική κυκλοφορία της το 1991. Πρόκειται για μια γλώσσα προγραμματισμού υψηλού πειπέδου, γενικής χρήσης. Ένα από τα σημαντικότερα χαρακτηριστικά της είναι η ευκολία ανάγνωσης του κώδικα, κάτι που ευκολύνει ιδιαίτερα την εκμάθηση, προγραμματισμό σε Python αλλά και συντήρηση μεγάλων προγραμμάτων γραμμένων σ'αυτήν.

Οι δυνατότητες της μπορούν να επεκταθούν σημαντικά μέσω των πρόσθετων (modules) και των βιβλιοθηκών (libraries). Θα γνωρίσουμε πρόσθετες δυνατότητες της Python σε επόμενο κεφάλαιο, όταν θα δημιουργήσουμε σχήματα με χρήση εντολών.

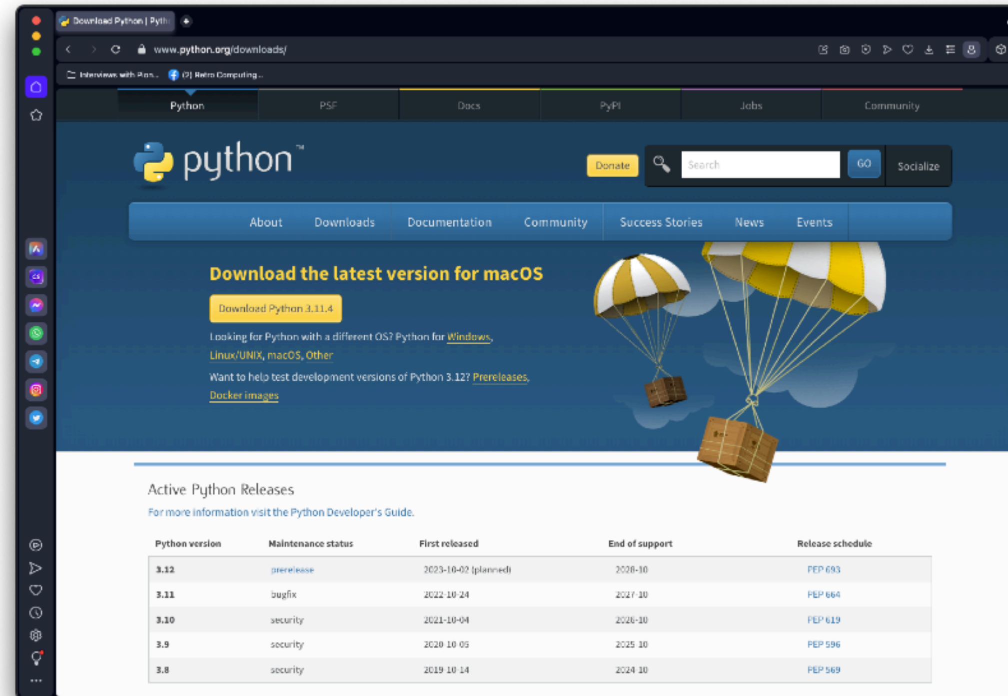


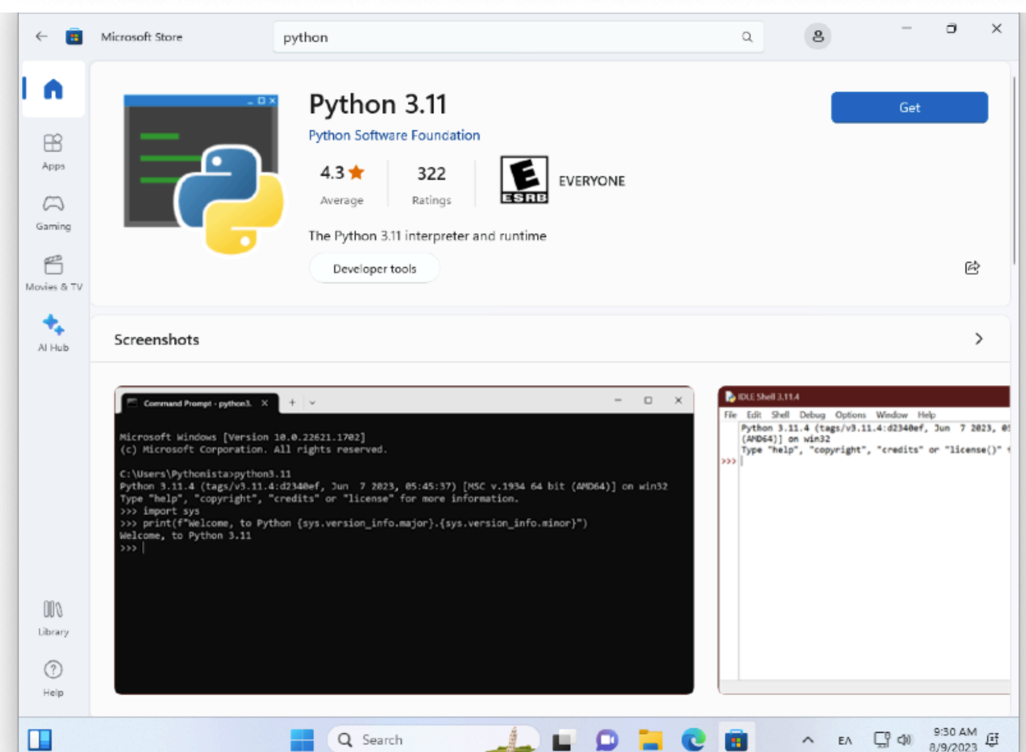
Εγκατάσταση Python

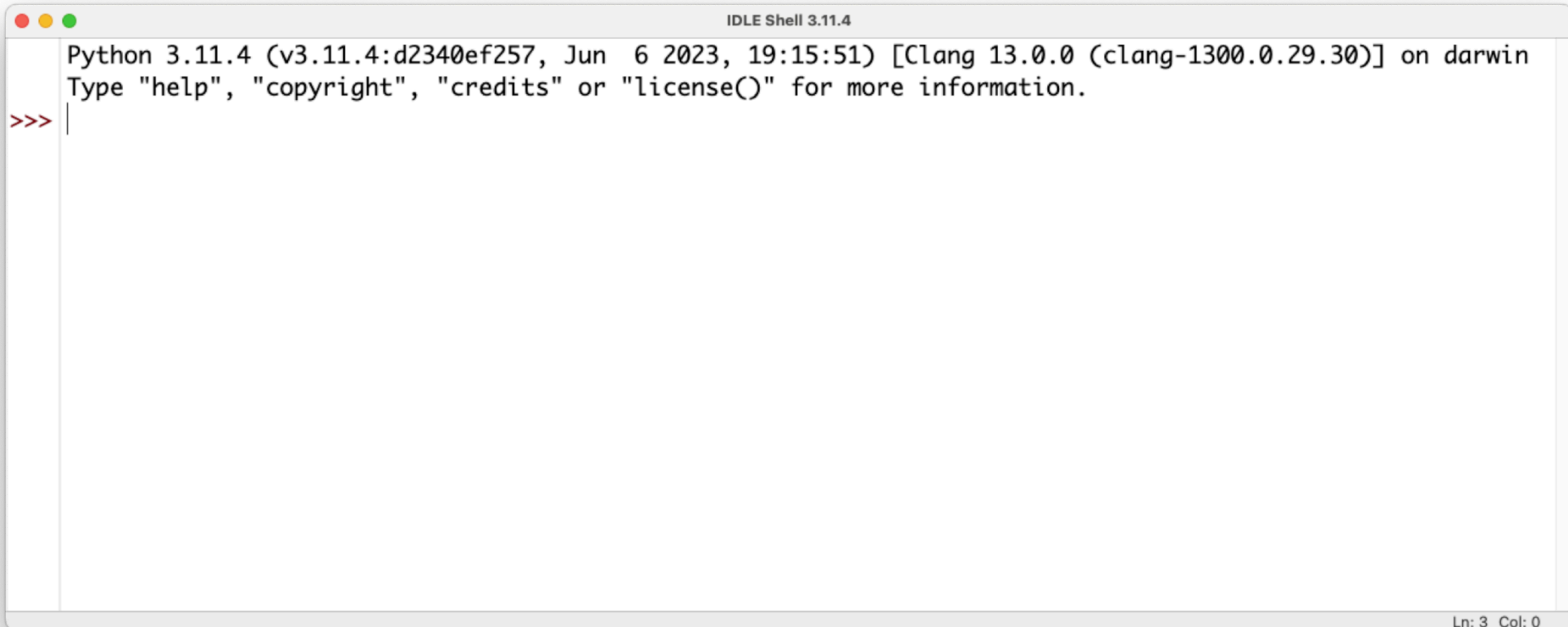
Ο πιο απλός τρόπος να προγραμματίσουμε σε Python, είναι με την εγκατάσταση των απαραίτητων πακέτων από την επίσημη σελίδα https://www.python.org/downloads/

Από εδώ μπορούμε να κατεβάσουμε την έκδοση για τον δικό μας υπολογιστή (μη σας αγχώνει, η ιστοσελίδα θα εντοπίσει αυτόματα τον τύπο υπολογιστή που χρησιμοποιείτε). Θα πρέπει να τονίσουμε πως η Python και όλα τα σχετικά της πακέτα, καθώς και οδηγοί εκμάθησης και χρήσης (της ιστοσελίδας) είναι εντελώς δωρεάν.

Αφού κατεβάσουμε το αρχείο, εκτελούμε τη διαδικασία εγκατάστασης. Στα Windows, μπορούμε να εγκαταστήσουμε την Python απευθείας από το Microsoft Store (εικόνα κάτω δεξιά).







Το περιβάλλον προγραμματισμού IDLE Shell 3.11.

Η πιο πάνω εικόνα δείχνει το περιεχόμενο του περιβάλλοντος προγραμματισμού της Python. Εμφανίζεται στην πρώτη γραμμή η έκδοση της Python που χρησιμοποιούμε, καθώς και το περιβάλλον του λειτουργικού συστήματος. Στην τελευταία γραμμή εμφανίζονται (στο περιθώριο) τα σύμβολα >>>. Δίπλα από αυτά μπορούμε να πληκτρολογήσουμε τις εντολές μας σε Python. Στη συνέχεια θα γνωρίσουμε βασικές εντολές της Python.

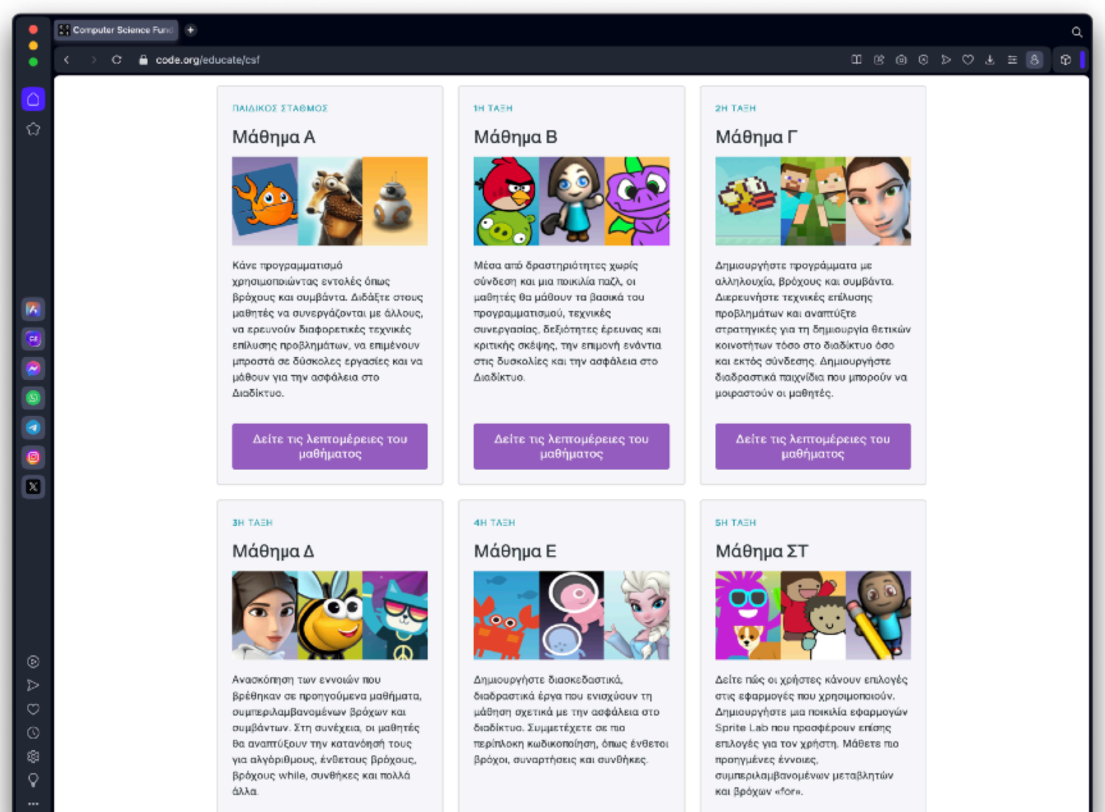
Τι μάθαμε μέχρι τώρα

- Στο κεφάλαιο αυτό μάθαμε για τις γλώσσες προγραμματισμού, που επιτρέπουν την επικοινωνία του ανθρώπου με τον υπολογιστή.
- Γνωρίσαμε γλώσσες και περιβάλλοντα προγραμματισμού που χρησιμοποιούνται στην εκπαίδευση.
- Μάθαμε την ιστορία της γλώσσας Python
- Εγκαταστήσαμε την Python στον υπολογιστή μας.

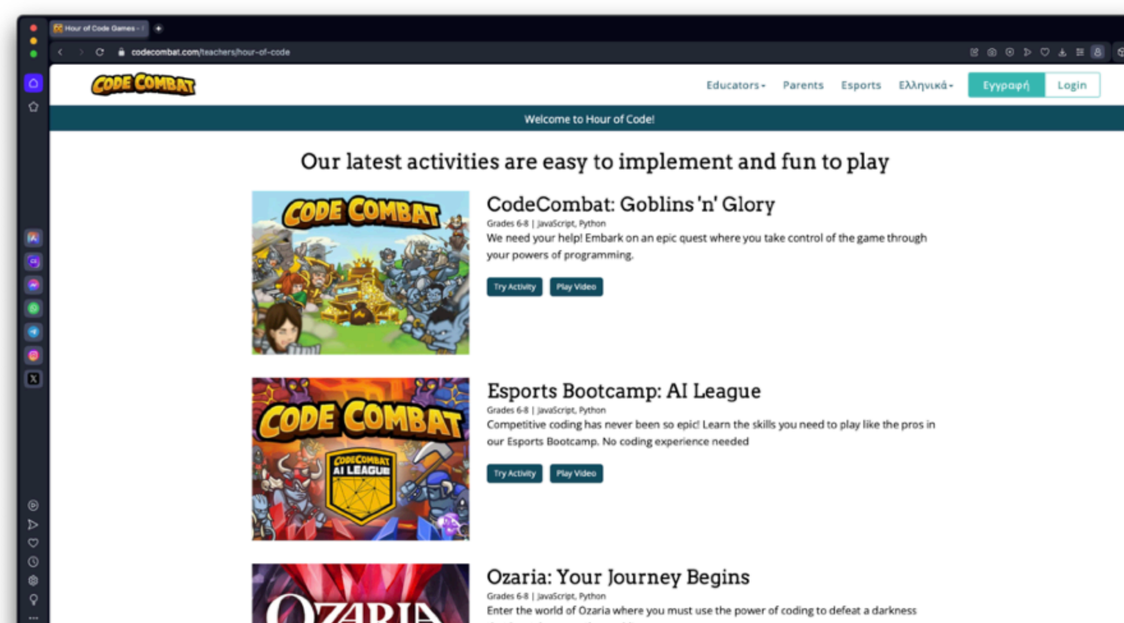


Δραστηριότητες

- Δοκιμάστε το Scratch Jr ή/και το Scratch. Και τα δύο μπορείτε να τα εγκαταστήσετε στο tablet ή στον υπολογιστή σας μέσω Google/Play Store (ή Windows Store), ενώ το Scratch "τρέχει" και διαδικτυακά: https://scratch.mit.edu
- Μια καλή εισαγωγή στον προγραμματισμό είναι οι δραστηριότητες του code.org. Δοκιμάστε τις εισαγωγικές δραστηριότητες από τον σύνδεσμο: https://code.org/educate/csf#pick-a-course



- All Can Code: ένα διαδικτυακό παιχνίδι εισαγωγής στον προγραμματισμό. Μπορείτε να το δοκιμάσετε από τη διεύθυνση: https://runmarco.allcancode.com
- CodeCombat: ένα διαδικτυακό παιχνίδι ρόλων που βασίζεται σε Python (και άλλες γλώσσες) για να δίνουμε οδηγίες στους χαρακτήρες που ελέγχουμε. Μέσα από μια σειρά από στάδια, γνωρίζουμε τη σύνταξη (έστω σε απλοποιημένη μορφή) εντολών σε Python. Η έκδοση "Hour of Code" μας επιτρέπει να δοκιμάσουμε δραστηριότητες και από το παιχνίδι Ozaria. https://codecombat.com/teachers/hour-of-code



ΜΕΡΟΣ Β΄: ΠΥΘΩΝΕΣ & ΚΩΔΙΚΑΣ



2. Γνωριμία με την Python

```
10 |  print (“I don’t think”)
20 |  print (“Then you shouldn’t talk”)
30 | #Alice in Wonderland
```

Πρώτες εντολές

Σίγουρα έχετε αγωνία να δώσετε τις πρώτες σας εντολές! Πριν γίνει αυτό, θα πρέπει να εξηγήσουμε πως, για τις επόμενες σελίδες, θα δουλεύουμε σε "διαδραστική" μορφή λειτουργίας. Τι σημαίνει αυτό; Η Python θα εκτελεί κάθε εντολή που πληκτρολογούμε, μία κάθε φορά. Αυτό σίγουρα δεν είναι βολικό αν θέλουμε να δημιουργήσουμε ένα πρόγραμμα με αρκετές εντολές. Είναι όμως ιδανικός τρόπος για να ξεκινήσουμε και να γνωρίσουμε εντολές της Python, καθώς και τη λογική του προγραμματισμού.

Θα **πρέπει όμως να προσέξουμε**: οι γλώσσες προγραμματισμού αναγνωρίζουν τις εντολές ακριβώς όπως πρέπει να είναι γραμμένες. Με λίγα λόγια, αν κάνουμε ορθογραφικά λάθη σε μια εντολή, τότε το πρόγραμμα μας δε θα την εκτελέσει!



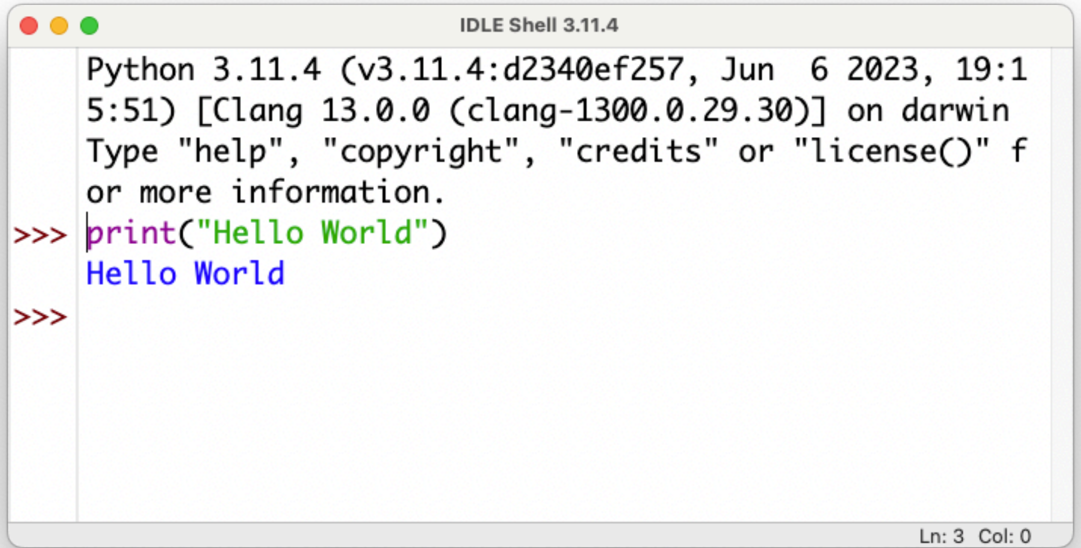
Τι θα γνωρίσουμε:



Στο **Κεφάλαιο 2: "Εισαγωγή στον Προγραμματισμό"**, θα γνωρίσουμε:

- Το διαδραστικό περιβάλλον της Python, όπου εκτελείται μια εντολή κάθε φορά
- Τη χρήση της print() για εμφάνιση πληροφοριών στην οθόνη (κειμένου και αριθμών)
- Τη χρήση της Python ως αριθμομηχανής
- Τις μεταβλητές, τα είδη των μεταβλητών και πώς τις χρησιμοποιούμε
- Τις λίστες, και πώς τις χρησιμοποιούμε, καθώς και τις διαφορές των λιστών από τις μεταβλητές
- Τη συνθήκη "if…" και λήψη αποφάσεων με αυτήν
- Εισαγωγή δεδομένων με το πληκτρολόγιο με την input()





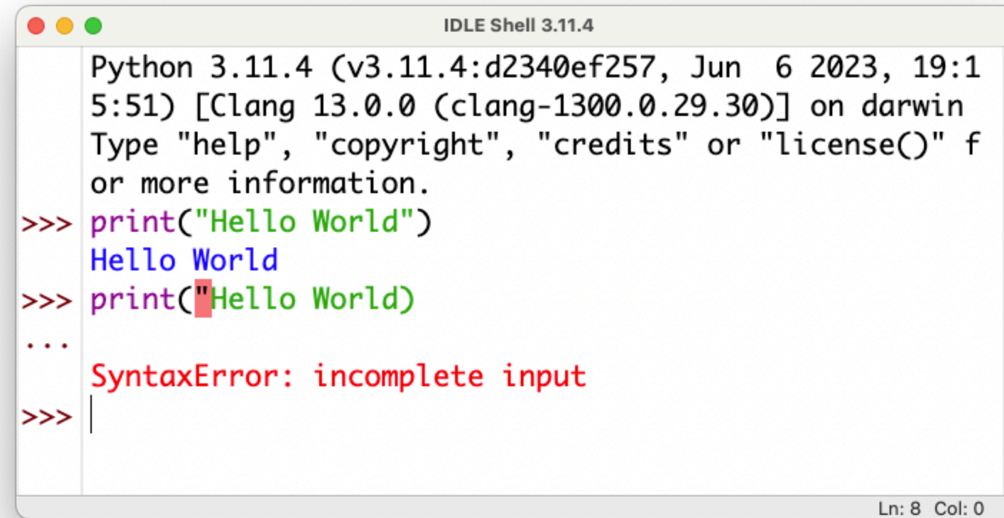
Ας δούμε την πρώτη μας εντολή - προσέξτε πως το περιβάλλον προγραμματισμού χρωματίζει τις εντολές.

Η εντολή `print( )` δεν εκτυπώνει κάτι σε χαρτί. Εμφανίζει όμως μηνύματα στην οθόνη μας. Αυτό που εμφανίζει είναι το μήνυμα `“Hello World”.` Με λίγα λόγια, οτιδήποτε βάλουμε στην παρένθεση, όταν πατήσουμε το πλήκτρο ENTER στο πληκτρολόγιό μας, θα εμφανιστεί στην οθόνη.

Θα παρατηρήσατε πως το `Hello World` δεν εμφανίζεται με εισαγωγικά. Και πολύ σωστά δεν εμφανίζεται, καθώς τα εισαγωγικά χρησιμεύουν μόνο για να “πουν” στην Python ποιο κείμενο να εμφανίσει στην οθόνη.

Θα δοκιμάσουμε τώρα το ίδιο μήνυμα, αλλά δεν θα κλείσουμε τα εισαγωγικά. Για να δούμε τί θα συμβεί…

Προσέξτε τις διαφορές: η παρένθεση, στη δεύτερη περίπτωση, εμφανίζεται με πράσινο και όχι μαύρο χρώμα.



Επειδή δεν κλείσαμε τα εισαγωγικά, η Python θεωρεί πως συνεχίζουμε να πληκτρολογούμε κείμενο για εμφάνιση στην οθόνη. Όταν πατήσαμε το πλήκτρο ENTER στο πληκτρολόγιο, η εντολή δεν εκτελέστηκε επειδή υπάρχει πρόβλημα στη σύνταξη της!

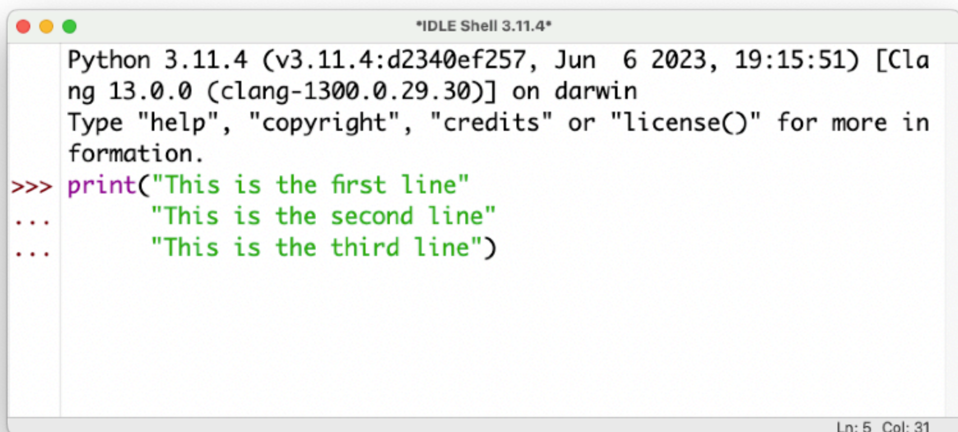
Η print() μέχρι τώρα:



Με την print() μπορούμε να εμφανίσουμε κείμενο (ή σειρές χαρακτήρων) πάνω στην οθόνη. Για να εκτελεστεί η εντολή, θα πρέπει, μέσα στην παρένθεση, το κείμενο μας να είναι σε εισαγωγικά.

```
>>> print(“This is just a text”)
```

Όπως έχουμε δει, κάθε εντολή εκτελείται με το πάτημα του ENTER στο πληκτρολόγιο. Αν θέλουμε να εμφανίσουμε περισσότερες γραμμές κειμένου, ο απλούστερος τρόπος είναι να μην κλείσουμε την παρένθεση. Πατάμε το ENTER στο τέλος της γραμμής και πληκτρολογούμε την επόμενη φράση/γραμμή. Κάθε φορά που πατάμε το ENTER (χωρίς να έχουμε κλείσει την παρένθεση), απλά αλλάζει η γραμμή χωρίς να εκτελεστεί η εντολή.



Στην εικόνα πιο πάνω, προσέξτε τις 3 τελείες στο πλαίσιο - αυτό δείχνει πως η εντολή δεν έχει ακόμη εκτελεστεί. Όταν όμως κλείσουμε την παρένθεση, η εντολή έχει ολοκληρωθεί και -με το πάτημα του ENTER- θα εκτελεστεί.

Στην εκτέλεση, δε θα εμφανιστεί η μια γραμμή κάτω από την άλλη. Θα εμφανιστεί η μια μετά την άλλη:

```
This is the first lineThis is the second lineThis is the third line
```

Αργότερα θα δούμε πώς μπορούμε να εμφανίσουμε κείμενο σε διαφορετικές γραμμές.

Εργασία με αριθμούς

Μέχρι τώρα χρησιμοποιήσαμε την εντολή print() για να εμφανίσουμε στην οθόνη κείμενο. Ας δούμε τι θα συμβεί αν, αντί για κείμενο, χρησιμοποιήσουμε αριθμούς.

```
>>> print(1234)
1234
>>> print("1234")
1234
```

Δείτε το παράδειγμα πιο πάνω: στην πρώτη γραμμή, μέσα στην παρένθεση είχαμε τον αριθμό 1234 χωρίς εισαγωγικά. Στην εκτέλεση, εμφανίστηκε ο αριθμός της παρένθεσης.

Στη δεύτερη εντολή, το “1234” ήταν σε εισαγωγικά. Αμέσως εμφανίστηκε με πράσινο χρώμα. Όταν εκτελέστηκε η εντολή, εμφανίστηκε και πάλι ο αριθμός 1234. Το αποτέλεσμα φαίνεται να είναι το ίδιο, όμως υπάρχει μεγάλη διαφορά, κάτι που θα δούμε στη συνέχεια!

Ας δούμε ξανά το παράδειγμα, αλλά αυτή τη φορά ας χρησιμοποιήσουμε το σύμβολο της πρόσθεσης.

Στην πρώτη γραμμή, όπως φαίνεται και στην εικόνα πιο κάτω, έχουμε την αριθμητική πρόταση σε εισαγωγικά. Η Python εμφανίζει στην οθόνη οτιδήποτε υπάρχει μέσα στα εισαγωγικά, όπως ακριβώς το έχουμε πληκτρολογήσει.

```
>>> print("12+5")
12+5
>>> print(12+5)
17
```

Στη δεύτερη εντολή, που δεν έχουμε χρησιμοποιήσει εισαγωγικά, η Python θεωρεί πως θέλουμε να εμφανίσουμε στην οθόνη, όχι την πράξη 12+5, αλλά το αποτέλεσμα της πράξης. Έτσι, μας εμφανίζει το άθροισμα του 12 συν 5, δηλαδή το 17.



Προτεραιότητα πράξεων:

Η Python εκτελεί τις εντολές με τη σωστή σειρά (προτεραιότητα πράξεων). Πρώτα πράξεις σε παρενθέσεις, στη συνέχεια πολλαπλασιασμοί και διαιρέσεις και τέλος προσθέσεις και αφαιρέσεις.

```
>>> print((10-2)*2)
```


Ας δούμε κάποια παραδείγματα πράξεων:

Πρόσθεση: `print(100 + 5)`

Αφαίρεση: `print(100 - 5)`

Πολλαπλασιασμός: `print(100 * 5)`

Διαίρεση: `print(100 / 5)`

Δυνάμεις: `print(5**100)`

Αν θέλουμε να κάνουμε πιο ενδιαφέρον το αποτέλεσμα που εμφανίζεται στην οθόνη, μπορούμε να συνδυάσουμε τη μαθηματική πρόταση με το αποτέλεσμα της πράξης.

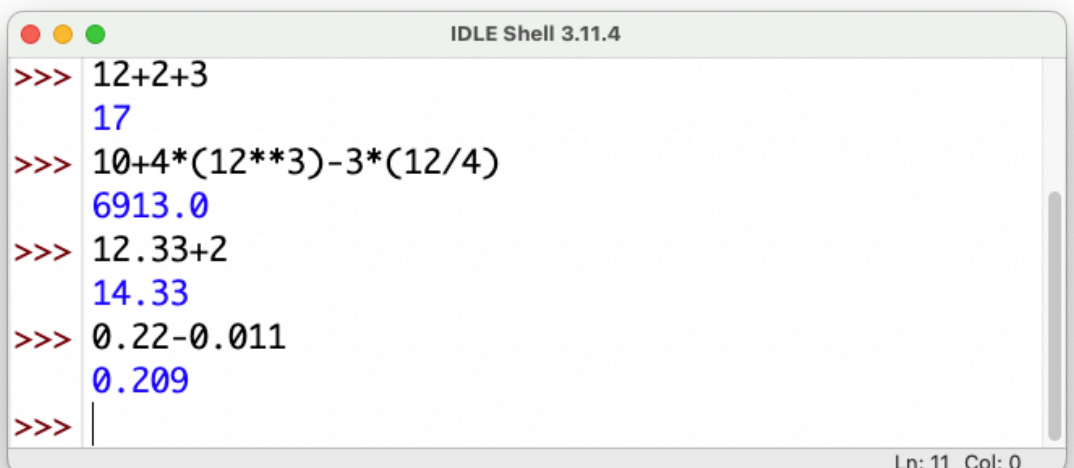
```
>>> print("12+5=",12+5)
```

Όταν εκτελεστεί η πιο πάνω εντολή, θα εμφανιστεί στην οθόνη το 12+5= που βρίσκεται μέσα στα εισαγωγικά. Στη συνέχεια, υπάρχει το κόμμα (,) που δηλώνει πως υπάρχει και άλλο τμήμα εντός της παρένθεσης που πρέπει να εκτελεστεί. Επειδή το μέρος που ακολουθεί είναι αριθμοί εκτός εισαγωγικών, η Python θα κάνει την πράξη και θα εμφανίσει το αποτέλεσμα:

```
12+5= 17
```

Python, η αριθμομηχανή!

Η Python μπορεί να χρησιμοποιηθεί για εκτέλεση μαθηματικών πράξεων. Στις προηγούμενες σελίδες είδαμε τον τρόπο λειτουργίας της Python, καθώς και τη χρήση της εντολής print(). Στην πραγματικότητα, δε χρειάζεται η συγκεκριμένη εντολή για να εκτελεστούν οι πράξεις. Μπορούμε απευθείας να δώσουμε τις πράξεις που θέλουμε, όπως στην εικόνα πιο κάτω.



Θα πρέπει να προσέξουμε μόνο τον τρόπο με τον οποίο γράφουμε τους δεκαδικούς αριθμούς: στην Python χρησιμοποιούμε πάντοτε το σύμβολο "." για διαχωρισμό του ακεραίου μέρους από το δεκαδικό. Αν χρησιμοποιήσουμε το σύμβολο "," η Python διαχωρίζει τους αριθμούς και προσθέτει σε ζευγάρια.

Ας δούμε ένα παράδειγμα:

```
>>>1,2 + 3,8
 (1, 5, 5)
```

Αν δούμε την πιο πάνω πράξη, η Python εκτέλεσε πρόσθεση στα ψηφία που βρίσκονται μεταξύ του συμβόλου "+" και ξεχώρισε το πρώτο ψηφία (1) που ήταν αριστερά του "," και το τελευταίο ψηφίο (8) που ήταν δεξιά από το ",".

Στην Python πάντοτε χρησιμοποιούμε το σύμβολο "." για να γράψουμε δεκαδικούς αριθμούς. Στις πράξεις με δεκαδικούς, πάντοτε η πρόσθεση γίνεται σύμφωνα με τη θέση του κάθε ψηφίου (μονάδες με μονάδες, δέκατα με τα δέκατα, εκατοστά με εκατοστά κοκ).

Στη συνέχεια θα γνωρίσουμε τις μεταβλητές, και αργότερα τις συναρτήσεις, που θα μας επιτρέψουν να γράψουμε πιο σύνθετο κώδικα. Επίσης θα χρησιμοποιήσουμε μαθηματικά για να εκτελέσουμε επαναλήψεις, κάτι που θα δούμε σε επόμενα κεφάλαια.



Εισαγωγή στις μεταβλητές

Μέχρι τώρα, γνωρίσαμε τον τρόπο προβολής κειμένου και αριθμών στην οθόνη. Μάθαμε επίσης πώς να χρησιμοποιήσουμε την Python για να εκτελέσουμε μαθηματικές πράξεις. Τα πράγματα γίνονται πολύ πιο ενδιαφέροντα, όταν γνωρίσουμε τις μεταβλητές!

Το ερώτημα είναι "τι είναι η μεταβλητή"; Θα πρέπει να φανταστούμε τις μεταβλητές ως "κουτάκια" στη μνήμη του υπολογιστή στα οποία τοποθετούμε (ή αφαιρούμε) αντικείμενα. Ένα αντικείμενο μπορεί να είναι ένα χαρτάκι με ένα όνομα. Ή ένας αριθμός. Ή μια φράση. Αυτό που πρέπει να ξέρουμε είναι πως, η μεταβλητή μπορεί να κρατήσει ένα πράγμα κάθε φορά!

Για να δημιουργήσουμε μια μεταβλητή, απλά την ονομάζουμε και της δίνουμε μια "τιμή" (περιεχόμενο). Για παράδειγμα:

```
>>> name="George"
```

Όταν πατήσουμε το ENTER, δε φαίνεται να συμβαίνει κάτι (δεν εμφανίζεται κάποιο μήνυμα στην οθόνη). Στην πραγματικότητα, έχουμε δημιουργήσει μια μεταβλητή που την ονομάσαμε name (θα μπορούσαμε να την είχαμε ονομάσει οτιδήποτε, ακόμη και SuperMarios). Στη συνέχεια, με το σύμβολο = δίνουμε μια τιμή (ένα περιεχόμενο) στη μεταβλητή. Επειδή το περιεχόμενο που θα δώσουμε είναι κείμενο, αυτό μπαίνει σε εισαγωγικά.

Στη συνέχεια πληκτρολογούμε και την εντολή:

```
>>> print(name)
```

Στην πιο πάνω εντολή, δε χρειάζεται να βάλουμε το name σε εισαγωγικά, γιατί είναι μεταβλητή. Στην ουσία ζητάμε από την Python να εμφανίσει στην οθόνη το περιεχόμενο της μεταβλητής name.

Στην οθόνη εμφανίζεται η λέξη George (το περιεχόμενο της μεταβλητής name)

```
George
```

Αν δεν χρησιμοποιούσαμε εισαγωγικά στο "George", τότε η Python θα θεωρούσε πως, μέσα στη μεταβλητή name θέλουμε να βάλουμε το περιεχόμενο μιας άλλης μεταβλητής, με το όνομα George. Επειδή (προς το παρόν) μια τέτοια μεταβλητή δεν υπάρχει, εμφανίζεται μήνυμα λάθους .

```
>>> name="Marios"
>>> name="Eleni"
>>> name="Kostas"
>>> print(name)
Kostas
```

Όπως αναφέραμε και στην προηγούμενη σελίδα, σε μια μεταβλητή μπορούμε να έχουμε μια πληροφορία κάθε φορά. Ας δούμε το πιο κάτω παράδειγμα:

Στην πρώτη γραμμή δημιουργούμε τη μεταβλητή name και της βάζουμε τη λέξη Marios. Στη δεύτερη γραμμή, στη μεταβλητή name, βάζουμε τη λέξη Eleni. Στην τρίτη γραμμή, στη μεταβλητή name, βάζουμε τη λέξη Kostas. Όταν εκτελέσουμε την εντολή print(name), εμφανίζεται η λέξη Kostas.



Κάθε φορά που δίνουμε μια νέα τιμή (ένα νέο περιεχόμενο) σε μια μεταβλητή, αντικαθιστούμε το προηγούμενο περιεχόμενο της με το νέο.

Μπορούμε να δουλέψουμε ταυτόχρονα με πολλές μεταβλητές. Στο παράδειγμα που ακολουθεί, έχουμε μια μεταβλητή για το όνομα και μια για το επίθετο. Με την εντολή print(name, surname) εμφανίζουμε μια γραμμή το περιεχόμενο και των δύο μεταβλητών. Προσέξτε το (,) που είναι απαραίτητο για να εμφανιστεί το περιεχόμενο περισσότερων από μία μεταβλητών στην ίδια εντολή.

```
>>> name="Alexis"
>>> surname="Ioannou"
>>> print(name,surname)
Alexis Ioannou
```

Με παρόμοιο τρόπο μπορούμε να δώσουμε αριθμητική τιμή σε μια μεταβλητή.

```
>>> arithmos=10
```

Στο πιο πάνω παράδειγμα, δημιουργήσαμε μια μεταβλητή με το όνομα arithmos και της δώσαμε την τιμή 10. Το 10 είναι αριθμός, έτσι δεν χρειάζεται εισαγωγικά. Ας δούμε τώρα πώς να χρησιμοποιήσουμε μεταβλητές σε μαθηματικές πράξεις:

```
>>> print(arithmos+5)

15
```

Με την εντολή print, ζητήσαμε να προσθέσει το περιεχόμενο της μεταβλητής arithmos (που είναι το 10) με τον αριθμό 5. Έτσι, το αποτέλεσμα είναι 15.

Σε παλαιότερες εποχές, ήταν συνηθισμένο οι εκπαιδευτικοί να βάζουν τιμωρίες τύπου “γράψε 100 φορές δε θα ξαναμιλήσω στο μάθημα”. Ευτυχώς σήμερα δεν υπάρχουν τέτοιες τιμωρίες! Αν όμως, την εποχή των παππούδων μας, υπήρχε η Python, τότε τα πράγματα για τους μαθητές θα ήταν πολύ πιο απλά. Θα μπορούσαν απλά να κάνουν το εξής:

```
>>> mytext=“Δε θα ξαναμιλήσω στο μάθημα”
```

Στη μεταβλητή mytext, έχουμε τοποθετήσει τη φράση “Δε θα ξαναμιλήσω στο μάθημα”. Τώρα θα δούμε πώς μπορούμε να την επαναλάβουμε με μια απλή εντολή:

```
>>> print(mytext*100)
```

Η εντολή πιο πάνω εμφανίζει το περιεχόμενο της μεταβλητής mytext, ενώ το σύμβολο * και ο αριθμός 100 επαναλαμβάνουν την εμφάνιση του στην οθόνη.

```
>>> print(keimeno*100)
δε θα ξαναμιλήσω στο μάθημαδε θα ξαναμιλήσω στο μάθημαδε θα ξαναμιλήσω στ
ο μάθημαδε θα ξαναμιλήσω στο μάθημαδε θα ξαναμιλήσω στο μάθημαδε θα ξαναμ
ιλήσω στο μάθημαδε θα ξαναμιλήσω στο μάθημαδε θα ξαναμιλήσω στο μάθημαδε
θα ξαναμιλήσω στο μάθημαδε θα ξαναμιλήσω στο μάθημαδε θα ξαναμιλήσω στο μ
άθημαδε θα ξαναμιλήσω στο μάθημαδε θα ξαναμιλήσω στο μάθημαδε θα ξαναμιλή
σω στο μάθημαδε θα ξαναμιλήσω στο μάθημαδε θα ξαναμιλήσω στο μάθημαδε θα
ξαναμιλήσω στο μάθημαδε θα ξαναμιλήσω στο μάθημαδε θα ξαναμιλήσω στο μάθη
μαδε θα ξαναμιλήσω στο μάθημαδε θα ξαναμιλήσω στο μάθημαδε θα ξαναμιλήσω
```

Οι μεταβλητές επίσης μπορούν να πάρουν την τιμή μιας άλλης μεταβλητής!

```
>>> firstnumber=10
```

Έχουμε δημιουργήσει μια μεταβλητή με το όνομα firstnumber και της έχουμε δώσει την τιμή 10

```
>>> secondnumber=firstnumber
```

Έχουμε δημιουργήσει μια μεταβλητή με το όνομα secondnumber και της έχουμε δώσει την τιμή που περιέχει η μεταβλητή firstnumber.

```
>>> print(secondnumber)
    10
```

Οι μεταβλητές μπορούν να πάρουν (σχεδόν) όποιο όνομα θέλουμε. Υπάρχουν εξαιρέσεις. Για παράδειγμα, δεν μπορούμε να χρησιμοποιήσουμε το “print” ως όνομα μεταβλητής. Το όνομα μπορεί να είναι και ένα γράμμα (π.χ. N=“School”). Όχι όμως αριθμός! Περισσότερα θα δούμε στη συνέχεια.



Εργασία με μεταβλητές

Οι πύθωνες είναι ιδιαίτερα έξυπνοι! Το ίδιο ισχύει και για την Python. Σε άλλες γλώσσες προγραμματισμού, χρειάζεται να πούμε από πριν αν μια μεταβλητή θα περιέχει κείμενο, ακέραιο αριθμό ή δεκαδικό. Η Python αναγνωρίζει από μόνη της το είδος της μεταβλητής.

Ας δούμε ένα παράδειγμα:

```
>>> myaddress="20, Ioannou Str"
```

Στην πιο πάνω μεταβλητή (myaddress), έχουμε κείμενο και αριθμό (η οδός και ο αριθμός της). Μεταβλητές με παρόμοιο περιεχόμενο, είναι τύπου String (συμβολοσειρά).

```
>>> myage=48
```

Στην πιο πάνω μεταβλητή (mage), δίνουμε ως τιμή έναν ακέραιο αριθμό. Αυτές οι μεταβλητές είναι τύπου Integer (ακέραιος)

```
>>> myheight=1.82
```

Στην πιο πάνω μεταβλητή (myheight), δίνουμε ως τιμή έναν δεκαδικό αριθμό. Αυτές οι μεταβλητές είναι τύπου Float (κινητής υποδιαστολής).

Κεφαλαία & Μικρά!

Οι μεταβλητές μπορούν να πάρουν διάφορα ονόματα, όπως είδαμε και σε προηγούμενη σελίδα. Θα πρέπει να προσέξουμε τα ονόματα τους να έχουν νόημα. Αυτό διευκολύνει πολύ την ανάγνωση του κώδικα, ώστε να καταλάβουμε ποιες μεταβλητές χρησιμεύουν για ποιο σκοπό.

Αν, για παράδειγμα, θέλουμε να έχουμε μια μεταβλητή που να κρατά το σκορ σε ένα παιχνίδι, θα ήταν πιο εύκολο να την ονομάσουμε "score" ή "myscore" (ή κάτι παρόμοιο). Θα μπορούσαμε απλά να την ονομάζαμε και S ή S2 (αν θα έχουμε διαφορετικά σκορ).

Όταν ονομάζουμε μεταβλητές, θα πρέπει επίσης να προσέξουμε τη χρήση κεφαλαίων χαρακτήρων. Δείτε το ακόλουθο παράδειγμα:

```
>>> myage=48

>>> myAge=24
```

Στο παράδειγμα πιο πάνω, φαίνεται να έχουμε την ίδια μεταβλητή (myage). Όμως, στη δεύτερη γραμμή, το A το γράψαμε με κεφαλαία γράμματα. Για την Python, είναι δύο διαφορετικές μεταβλητές!

Τώρα που γνωρίσαμε καλύτερα τις μεταβλητές, ας δούμε περισσότερα μαθηματικά με αυτές. Ήδη είδαμε παραδείγματα όπου πολλαπλασιάσαμε μεταβλητή που περιέχει κείμενο με ακέραιο αριθμό.

```
>>> programlang="Python"

>>> print(programlang*4)

    PythonPythonPythonPython
```

Τι συμβαίνει όταν προσθέσουμε έναν ακέραιο με μια μεταβλητή που περιέχει κείμενο;

```
>>> print(programlang+5)
```

Όταν πατήσουμε το ENTER για να εκτελεστεί η πιο πάνω εντολή, παίρνουμε το ακόλουθο μήνυμα:

```
Traceback (most recent call last):
  File "<pyshell#18>", line 1, in
  <module>
      print(programlang+5)

TypeError: can only concatenate str
(not "int") to str
```

Δεν μπορούμε να προσθέσουμε **ακέραιο (int)** σε μεταβλητή που περιέχει **κείμενο (str)**!

Ας δούμε όμως κάτι διαφορετικό:

```
>>> programlang1="My favorite language is"

>>> programlang2="Python"
```

Στις πιο πάνω εντολές έχουμε δημιουργήσει δύο μεταβλητές. Ας δούμε τώρα πώς εμφανίζονται με την εκτέλεση:

```
>>> print(programlang1+programlang2)

    My favorite language isPython
```

Το σύμβολο "+" έχει ενώσει το περιεχόμενο των δύο μεταβλητών. Δεν έχει, όμως, αφήσει απόσταση ανάμεσα στο κείμενο της μίας μεταβλητής και στο κείμενο της άλλης μεταβλητής! Σε μια τέτοια περίπτωση, αν επιθυμούμε να υπάρχει απόσταση, χρησιμοποιούμε το "," αντί για το "+".

```
>>> print(programlang1,programlang2)

    My favorite language is Python
```



Μπορείτε να εξασκηθείτε και με άλλα σύμβολα (αφαίρεσης, διαίρεσης) ώστε να δείτε πώς συμπεριφέρεται η Python με τις μεταβλητές!

Εργασία με λίστες

Μια μέρα, που λέτε, αποφάσισε ο Πύθωνας να καλέσει φίλους για φαγητό. Το ψυγείο και τα ντουλάπια ήταν άδεια, έτσι έπρεπε να πάει στην υπεραγορά να ψωνίσει. Επειδή είναι ξεχασιάρης, έπρεπε με κάποιο τρόπο να θυμηθεί πολλά και διαφορετικά πράγματα για να αγοράσει. Έτσι έφτιαξε μια λίστα!

Οι μεταβλητές, όπως τις είδαμε σε προηγούμενες σελίδες, μπορούν να κρατήσουν μια πληροφορία κάθε φορά. Για παράδειγμα, η μεταβλητή myfood="banana" κρατά την τιμή (περιεχόμενο) "banana" και τίποτα άλλο.

Θα μπορούσε, βέβαια, να χρησιμοποιήσει μια μεταβλητή που να περιέχει στα εισαγωγικά όλα τα πράγματα που θέλει π.χ.

```
myfood="banana, strawberry, raspberry".
```



Το πρόβλημα είναι πως θα μπερδευτεί: θα ψάχνει να βρει ΕΝΑ κουτί στην υπεραγορά που να τα έχει όλα αυτά μέσα!

Άρα, καλές οι μεταβλητές, όμως αν θέλουμε να αποθηκεύσουμε πολλές (και διαφορετικές) πληροφορίες, ώστε να βρίσκουμε την καθεμιά ξεχωριστά, θα χρησιμοποιήσουμε λίστες!

Μια λίστα έχει την ακόλουθη μορφή:

```
>>> mylist=["banana","strawberry","ice cream"]
```

Δίνουμε ένα όνομα στη λίστα, όπως θα το δίναμε και στη μεταβλητή (λίστα "mylist") πιο πάνω. Στη συνέχεια, πληκτρολογούμε το "=" ώστε να δώσουμε το περιεχόμενο της λίστας. Αν θα είναι κείμενο (string),γράφουμε το κάθε αντικείμενο ξεχωριστά με εισαγωγικά. Μια λίστα, σε αντίθεση με μια μεταβλητή, έχει το περιεχόμενο της μέσα στα σύμβολα "[ ]".

Για να δούμε το περιεχόμενο της λίστας, πληκτρολογούμε την εντολή:

```
>>> print(my list)

['banana', 'cherry', 'strawberry']
```

Κάθε αντικείμενο σε μια λίστα έχει τη δική του σειρά: το πρώτο αντικείμενο βρίσκεται στη θέση 0, το δεύτερο στη θέση 1, το τρίτο στη θέση 2 κοκ. Σε επόμενο κεφάλαιο θα δούμε τη χρησιμότητα της λίστας και των θέσεων των αντικειμένων.



“Λίστες”, ‘Λίστες’, Λίστες!

Ας δούμε τις πιο κάτω λίστες:

```
mylist=[‘banana’,’icecream’,’strawberry’]

mylist=[“banana”,”icecream”,”strawberry”]
```

Και στις δύο περιπτώσεις, το αποτέλεσμα είναι το ίδιο - η λίστα mylist περιέχει τις ίδιες πληροφορίες σε μορφή string.

Σε περίπτωση που θέλουμε να χρησιμοποιήσουμε ακέραιους αριθμούς (integers) στη λίστα, θα πρέπει να έχει την εξής μορφή:

```
mynewlist=[10,14,23,44]
```

Για να δείξουμε το περιεχόμενο μιας λίστας, δε χρειάζεται απαραίτητα η εντολή print(). Η print() είναι χρήσιμη όταν θέλουμε να προβάλουμε και άλλες πληροφορίες μαζί με το περιεχόμενο της λίστας. Μπορούμε απλά να πληκτρολογήσουμε το όνομα της λίστας ώστε να εμφανιστεί το περιεχόμενο της:



```
>>> mynewlist
    [10,14,23,44]
```

Αν έχουμε δύο λίστες, μπορούμε να ενώσουμε το περιεχόμενο τους. Για παράδειγμα:

```
>>>  mylist1=[“banana”,”watermelon”]
>>>  mylist2=[“apple”,”cherry”]

>>>  mylist2=mylist2+mylist1
```

Στην τελευταία εντολή, βάζουμε στη λίστα mylist2 το περιεχόμενο που είχε (η mylist2) μαζί με το περιεχόμενο της mylist1. Αν πληκτρολογήσουμε mylist2, θα πάρουμε το ακόλουθο:

```
>>> mylist2
    [‘apple’, ‘cherry’, ’banana’, ‘watermelon’]
```



Μπορούμε να προσθέσουμε περιεχόμενο σε μια λίστα με το σύμβολο “+” και τα αντικείμενα μέσα σε []. Για παράδειγμα:

```
mylist2=mylist2+[“mango”]
```

Η λέξη mango θα προστεθεί στο τέλος της λίστας μας!

Αποφάσεις...

Οι πύθωνες είναι έξυπνα ζώα. Ένας πύθωνας ήθελε να αναβαθμίσει τον υπολογιστή του. Ήθελε να δει αν η μνήμη του ήταν μεγαλύτερη από 8GB. Σκέφτηκε “ΑΝ η μνήμη (του υπολογιστή) είναι μικρότερη από 8GB, θα την αναβαθμίσω”. Πριν πάρει την απόφαση να αναβαθμίσει τη μνήμη του υπολογιστή του, πρώτα θα έπρεπε να ελέγξει κατά πόσο ήταν λιγότερη από 8GB (και, ξέρετε, με λιγότερα από τόσα δεν τρέχετε γρήγορα Windows 11 ή MacOS X ή νέες εκδόσεις του Linux).

Πώς θα το ελέγχαμε (σχεδόν) με ψευδοκώδικα:

```
“Αν η μνήμη (του υπολογιστή) < 8GB
    αναβάθμισε μνήμη”
```

Άρα, πρώτα θα ελέγξουμε αν η μνήμη μας είναι μικρότερη (το σύμβολο “<”) από 8GB. Αν ισχύει αυτό, τότε και μόνο τότε θα αναβαθμίσουμε τη μνήμη του υπολογιστή!



Ας δούμε όμως σε κώδικα πώς δουλεύει αυτό:

```
>>> mymemory=8
```

Έχουμε δημιουργήσει μια μεταβλητή στην οποία δίνουμε την τιμή 8 (η μνήμη του υπολογιστή μας σε GB).

```
>>> if mymemory<8: print("Χρειάζεστε αναβάθμιση")
```

Η νέα εντολή "if" ζητά να ελέγξουμε αν ισχύει μια συνθήκη. Στην περίπτωση αυτή, ζητούμε να ελέγξει ΑΝ η τιμή της μεταβλητής είναι μικρότερη από 8. Σε περίπτωση που είναι μικρότερη από 8, τότε θα εμφανιστεί το μήνυμα Upgrade memory.

Στο παράδειγμα πιο πάνω, χρησιμοποιήσαμε μόνο μια συνθήκη (memory<8). Έτσι χρησιμοποιήσαμε μόνο μια γραμμή για τον κώδικα της συνθήκης.

Συνήθως στις συνθήκες ακολουθούμε την εξής δομή:

```
mymemory=8
if mymemory<8:
    print("Χρειάζεστε αναβάθμιση")
```

Το αποτέλεσμα βέβαια είναι το ίδιο, μιας και έχουμε και πάλι μία και μόνο συνθήκη να ελέγξουμε. Στη συνέχεια θα δούμε και άλλα παραδείγματα.

Μέχρι τώρα, ελέγχαμε έναν αριθμό και δίναμε μόνο μία απάντηση αν ίσχυε η συνθήκη. Αν όμως δεν ισχύει, γιατί να μη δίνουμε κάποια πληροφορία; Εδώ θα χρησιμοποιήσουμε την εντολή "else".

Ας δώσουμε την εξής εντολή:

```
print("Χρειάζεστε αναβάθμιση") if mymemory<8 else
print("Δεν χρειάζεστε αναβάθμιση")
```

Στο πιο πάνω παράδειγμα, ακολουθήσαμε διαφορετική σύνταξη. Σε μία γραμμή ζητάμε να εμφανιστεί στην οθόνη το μήνυμα Χρειάζεστε αναβάθμιση ΑΝ η τιμή της mymemory είναι μικρότερη από 8, ΔΙΑΦΟΡΕΤΙΚΑ (else) να εμφανιστεί το μήνυμα Δεν χρειάζεστε αναβάθμιση.

Τις συνθήκες θα τις μελετήσουμε και σε επόμενο κεφάλαιο, σε μεγαλύτερο βάθος.

Είναι σημαντικό να κατανοήσουμε πως, τα παραδείγματα μας μέχρι τώρα αφορούσαν την εφαρμογή μιας εντολής κάθε φορά. Αργότερα θα γνωρίσουμε τον τρόπο δημιουργίας ενός σύνθετου προγράμματος που θα αποτελείται από πολλές γραμμές κώδικα.

Στις προηγούμενες σελίδες, γνωρίσαμε τη εντολή "if" και μελετήσαμε κάποια απλά παραδείγματα. Είναι σημαντικό να δούμε, στη συνέχεια, τον τρόπο με τον οποίο μπορούμε να χρησιμοποιήσουμε την εντολή αυτή για πιο σύνθετες συγκρίσεις.

Θα ξεκινήσουμε αρχικά με δύο μεταβλητές:

```
number1=10
number2=20
```

Στις δύο αυτές μεταβλητές δώσαμε διαφορετικές τιμές.

Θέλουμε να συγκρίνουμε το περιεχόμενο των δύο μεταβλητών, ώστε να διαπιστώσουμε αν είναι το ίδιο.

```
if number1 == number2: print ("Equal")
```

Με την πιο πάνω εντολή, ελέγχουμε αν το περιεχόμενο τους είναι το ίδιο. Αυτό επιτυγχάνεται με το διπλό σύμβολο "=". Στο πιο πάνω παράδειγμα, η συνθήκη δεν ισχύει, έτσι δε θα εμφανιστεί κανένα μήνυμα.

Αν αυτό που μας ενδιαφέρει είναι να ελέγξουμε αν η τιμή τους είναι διαφορετική, τότε χρησιμοποιούμε την εντολή

```
if number1 != number2: print ("Equal")
```

Έλεγχοι που μπορούμε να κάνουμε:

- Μια μεταβλητή είναι μικρότερη από μια άλλη:
  `number1 < number2`
- Μια μεταβλητή είναι μεγαλύτερη από μια άλλη:
  `number1 > number2`
- Μια μεταβλητή είναι μικρότερη ή ίση από άλλη:
  `number1 <= number2`
- Μια μεταβλητή είναι μεγαλύτερη ή ίση από άλλη:
  `number1 >= number2`

Για συγκρίσεις, μπορούμε επίσης να χρησιμοποιήσουμε τα **And**, **Or** και **Not**.

```
if number1=100 and number2=100:
print("Equal")
```

Στην πιο πάνω εντολή ελέγχουμε αν ισχύουν και οι δύο συνθήκες (και ο ένας αριθμός, και ο άλλος, ίσοι).

```
If number1=100 or number2=100: print("One hundred")
```

Στην πιο πάνω εντολή, ελέγχουμε αν έστω ο ένας από τους δύο ισούται με 100.

Τι μάθαμε μέχρι τώρα

- Στο κεφάλαιο αυτό γνωρίσαμε γενικές πληροφορίες για τις γλώσσες προγραμματισμού και εστιάσαμε στη Python.
- Με την εντολή print() μπορούμε να εμφανίσουμε κείμενο και αποτέλεσμα πράξεων και μεταβλητών στην οθόνη. Το κείμενο πρέπει να είναι σε εισαγωγικά " ".
- Μαθηματικές πράξεις (π.χ. 5+7) εκτελούνται αυτόματα με το πάτημα του ENTER χωρίς τη χρήση της print().
- Οι μεταβλητές μπορούν να χρησιμοποιηθούν για να αποθηκεύσουν χαρακτήρες (string), ακέραιους (integers) και δεκαδικούς (float).
- Με το if μπορούμε να ελέγξουμε αν μια συνθήκη ισχύει ή όχι.
- Χρησιμοποιήσαμε λίστες και γνωρίσαμε τις διαφορές τους σε σχέση τις μεταβλητές.



Δραστηριότητες

1. Ποιο είναι το αποτέλεσμα της εκτέλεσης του πιο κάτω κώδικα;

```
number1=10
number2=5
name=input("Πώς σας λένε;")
print("Καλωσορίσατε",name)
number=5
print("Το άθροισμα των δύο αριθμών
είναι",number1+number)
```

2. Στον πιο κάτω κώδικα, (α) να αναφέρετε τις πληροφορίες που δεν είναι απαραίτητες, (β) να αναφέρετε το αποτέλεσμα της εκτέλεσης του:

```
numb1=10
numb2=4
numb3=2
numb4=8
#Πράξεις με τις πιο πάνω μεταβλητές
print("Το αποτέλεσμα των πράξεων
είναι",numb1+numb3+numb3*2)
```

3. Το πιο κάτω πρόγραμμα πρέπει να προσθέτει τις τιμές των τριών μεταβλητών, και να εμφανίζει το αποτέλεσμα τους. (α) Να συμπληρώσετε τον κώδικα που λείπει. (β) Να γράψετε το αποτέλεσμα της εκτέλεσης του κώδικα.

```
print("Πρόσθεση ακέραιων")
a=20
b=15
c=30
print
```

4. Να διορθώσετε τον πιο κάτω κώδικα. Στη συνέχεια, να γράψετε το αποτέλεσμα της εκτέλεσης του.

```
number1=5
number2=2
print("Το άθροισμα είναι, number1+number2)
print(Το γινόμενο είναι number1*number)
```

3. “Είναι πολλές οι εντολές”

```
10 |  print (“Who in the world am I?”)
20 |  print (“Ah, that’s the great puzzle!”)
30 | #Alice in Wonderland
```

Πρόγραμμα...

Οι πύθωνες είναι αρκετά έξυπνα ζώα, όπως έχουμε αναφέρει και στα προηγούμενα κεφάλαια! Μέχρι τώρα, εργαστήκαμε με μία εντολή κάθε φορά. Οι πύθωνες, όμως, μπορούν να εργαστούν με πολλές εντολές. Με αυτό τον τρόπο μπορούμε να δημιουργήσουμε πολύπλοκα προγράμματα, τα οποία όμως εκτελούν πολλές και διαφορετικές εργασίες.

Στο κεφάλαιο αυτό θα γνωρίσουμε νέες εντολές, τις οποίες και θα συνδυάσουμε για να δημιουργήσουμε το πρώτο μας πρόγραμμα!



Τι θα γνωρίσουμε:



- Στο **Κεφάλαιο 3 "Είναι πολλές οι εντολές"** θα εργαστούμε με κώδικα πολλών γραμμών.
- Θα χρησιμοποιήσουμε την print() για να εμφανίσουμε στην οθόνη, ταυτόχρονα, πολλές γραμμές με πληροφορίες.
- Με την εντολή input() θα δώσουμε από το πληκτρολόγιο τιμή σε μια μεταβλητή.
- Με την if...else θα ελέγξουμε αν ισχύει μία συνθήκη (π.χ. 10 > 5) ώστε να "αποφασίσει" το πρόγραμμά μας για την πορεία ενός απλού παιχνιδιού.
- Με την type() θα προβάλουμε τον τύπο μιας μεταβλητής.
- Με την βοήθεια της int(input()) θα εισαγάγουμε δεδομένα από το πληκτρολόγια ως ακέραιο αριθμό.







Μέχρι τώρα, γράφαμε και εκτελούσαμε μία εντολή κάθε φορά. Αυτό είναι χρήσιμο, όμως δε μας βοηθά όταν θέλουμε να δημιουργήσουμε μια σύνθετη εφαρμογή.

Στο πιο πάνω παράδειγμα, υπολογίσαμε το τετράγωνο του 5 (ή, το 5 στη δεύτερη δύναμη).

Για να μπορέσουμε να γράψουμε ένα πρόγραμμα (μια σειρά από εντολές) στην Python, μπορούμε να χρησιμοποιήσουμε ακόμη και απλές εφαρμογές συγγραφής κειμένου! Η εφαρμογή Notepad στα Windows ή το SimpleText σε MacOS είναι αρκετές για να γράψουμε τον κώδικα!

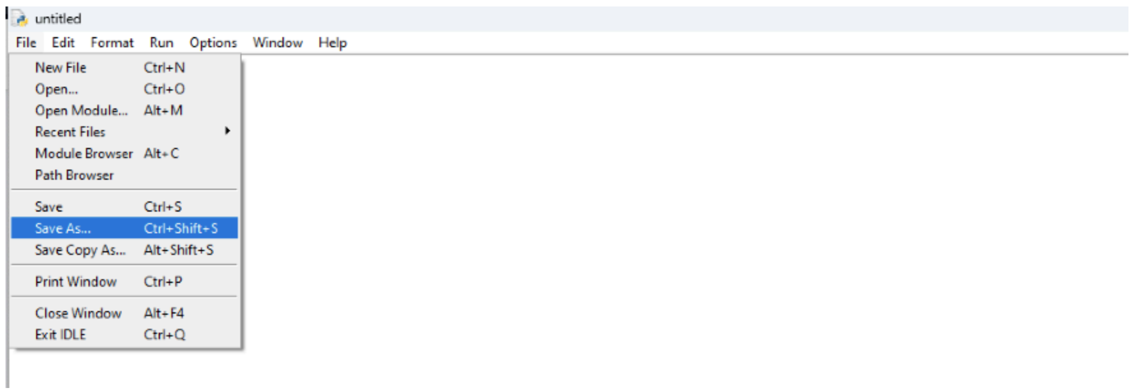
Για τη δημιουργία ενός προγράμματος με αρκετές γραμμές κώδικα, χρησιμοποιούμε λογισμικά επεξεργασίας κώδικα (Editors). Το IDLE περιλαμβάνει τον δικό του Editor. Από το μενού File του IDLE, επιλέγουμε New File (εικόνα πάνω δεξιά).

Παρατηρούμε αμέσως τις διαφορές ανάμεσα στο νέο παράθυρο, στο οποίο θα γράψουμε μια σειρά εντολών (πάνω δεξιά) με το παράθυρο του IDLE με το οποίο εργαστήκαμε στα προηγούμενα κεφάλαια.

Στην επόμενη σελίδα θα δημιουργήσουμε το πρόγραμμα μας και θα γνωρίσουμε και νέες εντολές.

Αρχεία Python

Πριν αρχίσουμε να προσθέτουμε εντολές, είναι καλό να αποθηκεύσουμε το αρχείο μας. Από το μενού File επιλέγουμε Save Us...



Ονομάζουμε το πρόγραμμα μας “myfirstpython.py” και το αποθηκεύουμε. Έχουμε κάνει ήδη ένα τεράστιο βήμα στην Python!

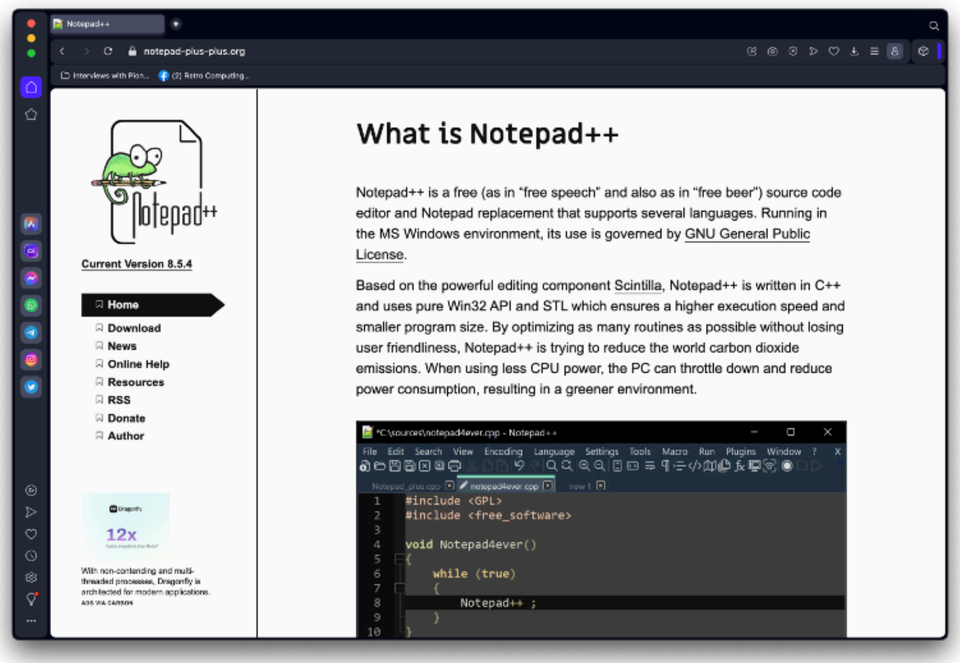


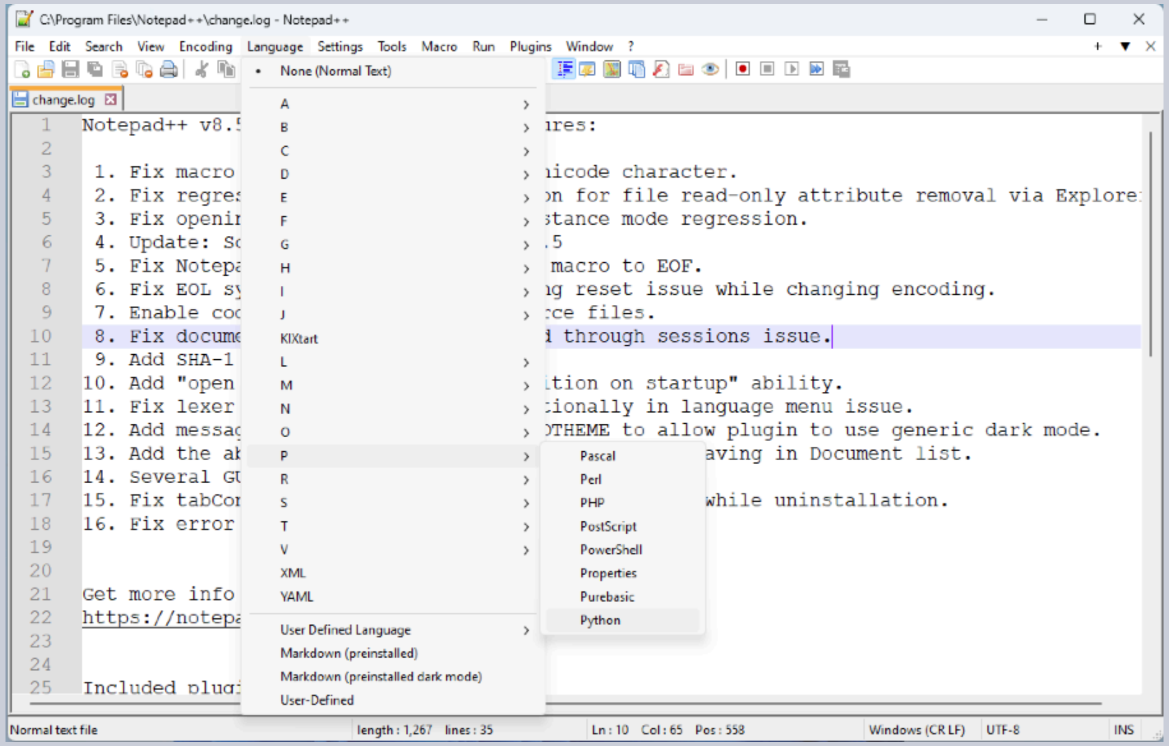
Τα αρχεία της Python έχουν κατάληξη .py, Το πρώτο μας αρχείο έχει ονομασία myfirstpython.py. Είναι καλό να δίνουμε στα αρχεία μας ονόματα που μας δείχνουν τι ακριβώς κάνει το πρόγραμμα.

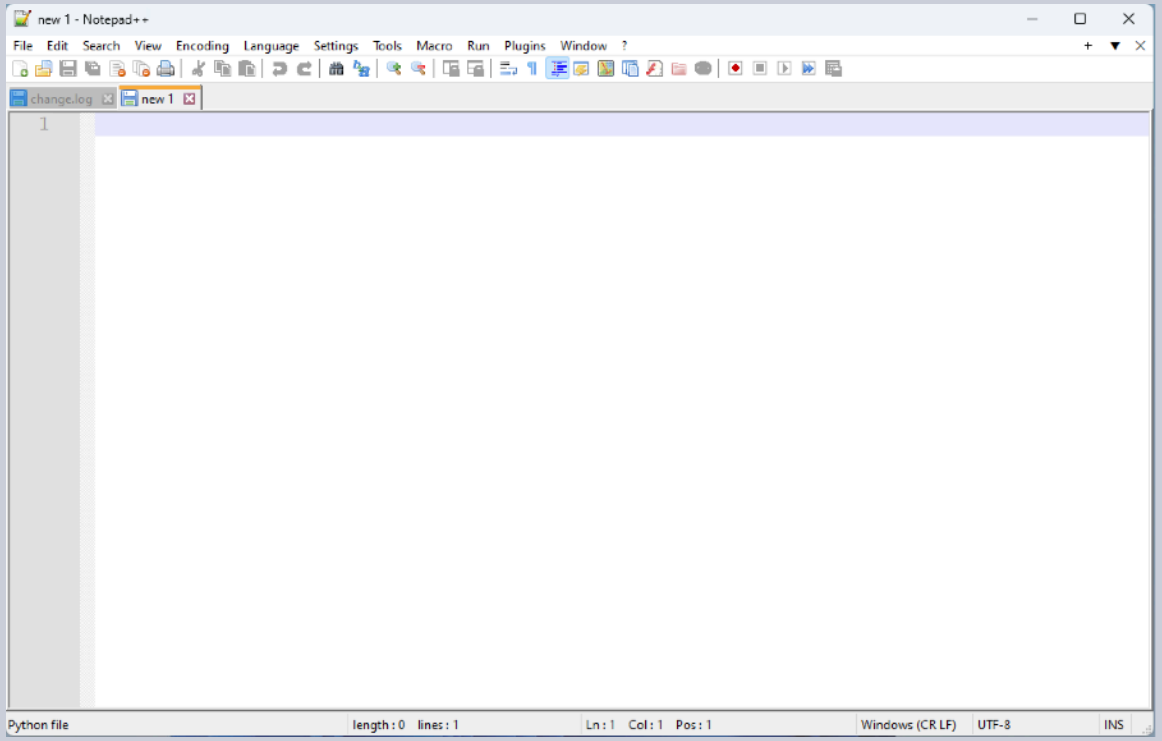
Εναλλακτικά...

Εκτός από το IDLE της Python, μπορούμε να χρησιμοποιήσουμε και άλλες εφαρμογές, οι οποίες δίνουν και άλλες δυνατότητες (για παράδειγμα, υποστήριξη πολλών γλωσσών προγραμματισμών).

Υπάρχουν πάρα πολλά λογισμικά που μπορούν να μας εξυπηρετήσουν. Σε MacOS, μια καλή επιλογή είναι το BBEDIT. Σε Windows, μια καλή επιλογή είναι το Notepad++ (εικόνα κάτω). Και τα δύο είναι δωρεάν και μπορούμε να τα κατεβάσουμε από το App Store του υπολογιστή μας ή από τις ιστοσελίδες τους.







Σχετικά με το Notepad++

Αν θέλετε να δοκιμάσετε κάτι διαφορετικό, μπορείτε να γράψετε κώδικα στο Notepad++, μια εξαιρετική δωρεάν εφαρμογή για Windows.

Με την εκκίνηση του Notepad++, εμφανίζονται γενικές πληροφορίες για το πρόγραμμα που κατεβάσαμε. Μας ενδιαφέρει να "πούμε" στο Notepad++ ότι θέλουμε να προγραμματίσουμε σε Python.

Όταν ορίσουμε τη Python ως την κύρια γλώσσα, τότε εμφανίζεται και ο αντίστοιχος χρωματισμός στις εντολές.

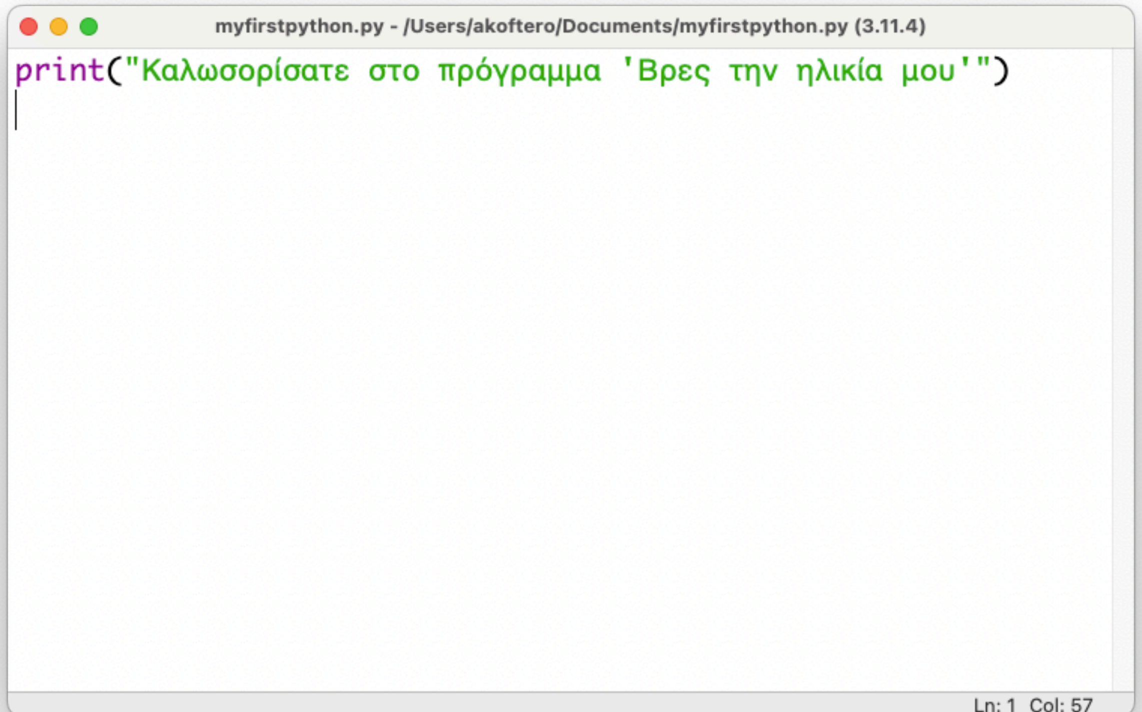
Από το μενού Language, επιλέγουμε Python (από το γράμμα "P"), όπως φαίνεται και στην εικόνα πάνω.

Για να δημιουργήσουμε ένα νέο αρχείο, από το μενού File επιλέγουμε New (εικόνα πάνω δεξιά).

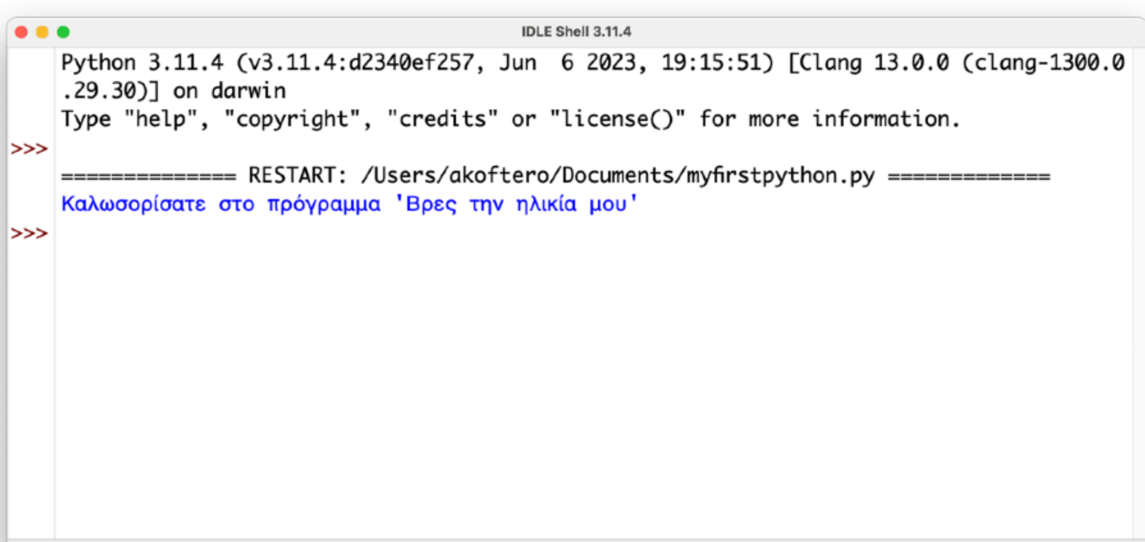
Το Notepad++ χρησιμοποιείται (και) από μαθητές/μαθήτριες και φοιτητές/φοιτήτριες, καθώς είναι δωρεάν και προσφέρει μεγάλη ευελιξία με υποστήριξη πολλών γλωσσών προγραμματισμού.



Let's Run!



Έχουμε γράψει την πρώτη μας εντολή! Σε αντίθεση με προηγούμενα παραδείγματα, με την αλλαγή γραμμής, δεν εκτελείται! Θα πρέπει, από το μενού Run, να επιλέξουμε "Run Module".



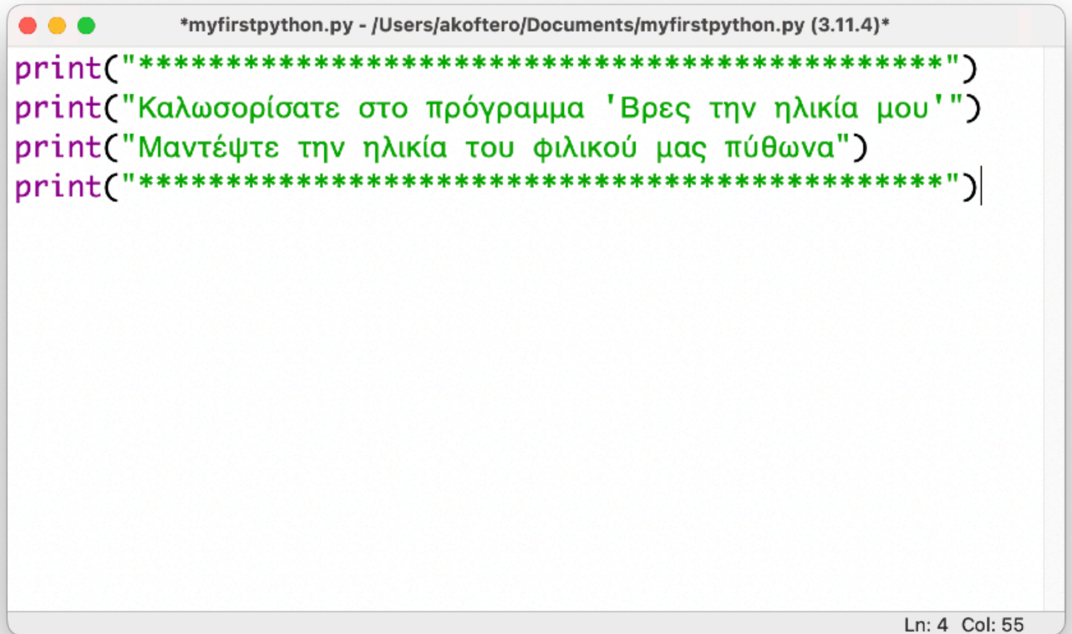


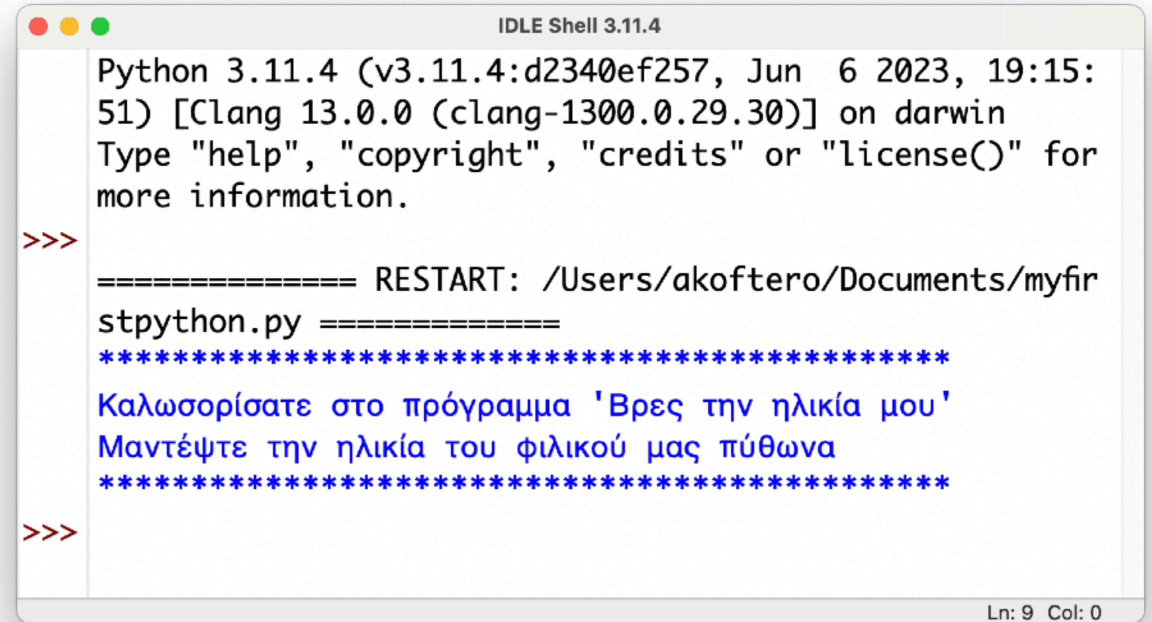
Το πρόγραμμα μας "τρέχει" και εμφανίζεται το αποτέλεσμα του (το μήνυμα Καλωσορίσατε στο πρόγραμμα 'Βρες την ηλικία μου').

Με την εντολή 'Run', το πρόγραμμα μας από Python (που καταλαβαίνουμε εμείς) "μεταφράζεται" στη γλώσσα που καταλαβαίνουν οι πύθωνες (οκ… οι υπολογιστές). Αυτό συμβαίνει κάθε φορά που επιλέγουμε το 'Run'



Στις επόμενες σελίδες θα εμπλουτίσουμε το πρόγραμμά μας με νέες εντολές!





Έχουμε προσθέσει 4 γραμμές εντολών print. Στην πρώτη και στην τέταρτη γραμμή, χρησιμοποιούμε το σύμβολο "*" για να εμφανιστεί πλαίσιο πάνω και κάτω από το κείμενο μας (εικόνα πάνω δεξιά).

Η γραμμή 2 και 3 του κώδικα μας (εικόνα πάνω) μας δίνει πληροφορίες για το πρόγραμμα. Στη γραμμή 2 μας δίνει το όνομα του παιχνιδιού, ενώ στη γραμμή 3 μας εξηγεί ποιος είναι ο σκοπός μας.



Το σύμβολο "*" μπορούμε να το χρησιμοποιήσουμε σε συνδυασμό με την εντολή print() για να δημιουργήσουμε σχήματα στην οθόνη. Θα δημιουργήσουμε σχήματα σε επόμενο κεφάλαιο αυτού του βιβλίου

Στη συνέχεια, θα προσθέσουμε και άλλο κώδικα.

```
print()
print()
print("Πόσα χρόνια ζει ένας πύθωνας;")
```

Οι δύο γραμμές print() που έχουμε προσθέσει, αφήνουν δύο γραμμές κενές ανάμεσα στο εισαγωγικό μέρος του προγράμματος, και στο ερώτημα που θέσαμε.

Στη συνέχεια, θα δείξουμε πώς δίνουμε πληροφορίες με το πληκτρολόγιο.



Εισαγωγή δεδομένων

Μέχρι τώρα, καταφέραμε να γράψουμε κώδικα 7 γραμμών. Όμως, το πρόγραμμά μας δεν μας επιτρέπει να δώσουμε την ηλικία που μπορεί να έχει ένας πύθωνας. Για να δώσουμε πληροφορίες από το πληκτρολόγιο, θα πρέπει να χρησιμοποιήσουμε μια νέα εντολή, την input.

```
age=input()
```

Ας δούμε την πιο πάνω εντολή: το “age” είναι μια μεταβλητή.Στη μεταβλητή αυτή θα αποθηκεύσουμε την ηλικία (αριθμό) που θα δώσουμε από το πληκτρολόγιο.

Η “input” είναι η εντολή με την οποία η Python περιμένει να δώσουμε μια πληροφορία με το πληκτρολόγιο. Μόλις πατήσουμε το πλήκτρο ENTER, η πληροφορία που πληκτρολογήσαμε (αριθμός, κείμενο) αποθηκεύεται στη μεταβλητή “age”.

```
print(“Πόσα χρόνια ζει ένας πύθωνας;”)
age=input()
```

Σημείωση: η μεταβλητή πιο πάνω δεν έχει τη σωστή μορφή! Αυτό θα το εξηγήσουμε αργότερα, όταν θα μιλήσουμε για αλφαριθμητικές τιμές και για ακέραιους!

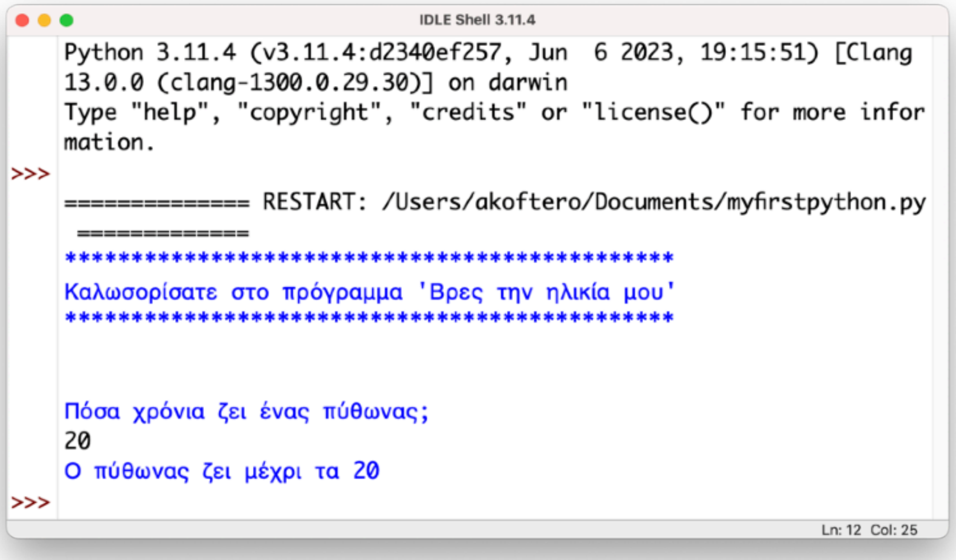
Όταν εκτελέσουμε το πιο πάνω πρόγραμμα, εμφανίζεται το μήνυμα της print (“Πόσα χρόνια ζει ένας πύθωνας;”) και αναμένει ο υπολογιστής να πληκτρολογήσουμε μια πληροφορία. Πατάμε ENTER για να ολοκληρωθεί η διαδικασία.

Για να κάνουμε πιο ενδιαφέρον το πρόγραμμα μας, προσθέτουμε ακόμη μια εντολή:

```
print(“Ζει μέχρι τα”,age)
```

Η πιο πάνω εντολή εμφανίζει στην οθόνη το μήνυμα “Ζει μέχρι τα” και ακολουθεί ο αριθμός που δώσαμε με το πληκτρολόγιο (και αποθηκεύθηκε στη μεταβλητή “age”).

Στη συνέχεια θα βελτιώσουμε ακόμη περισσότερο τον κώδικα, με χρήση και άλλων εντολών.



Ας δούμε ξανά την εντολή input():

```
print("Πόσα χρόνια ζει ένας πύθωνας;")
age=input()
```

Στο πιο πάνω παράδειγμα χρησιμοποιήσαμε δύο εντολές. Η print() μας δίνει το μήνυμα που εξηγεί τι πρέπει να πληκτρολογήσουμε. Στη δεύτερη εντολή, με την input() ζητάμε να πληκτρολογήσει κάποιος/α έναν αριθμό. Ο αριθμός αποθηκεύεται στη μεταβλητή age.

Θα μπορούσαμε τις δύο πιο πάνω να τις αντικαταστήσουμε με μία εντολή, που κάνει ακριβώς το ίδιο:

```
age=input("Πόσα χρόνια ζει ένας πύθωνας;")
```

Μέσα στις παρενθέσεις της input() έχουμε πληκτρολογήσει το μήνυμα (κείμενο) που είχαμε στην print().

```
****************************************
Καλωσορίσατε στο πρόγραμμα 'Βρες την ηλικία μου'
****************************************


Πόσα χρόνια ζει ένας πύθωνας;
```

Στο πιο πάνω παράδειγμα, το σημείο εισαγωγής του αριθμού είναι στα δεξιά του κειμένου (εικόνα κάτω αριστερά). Μπορούμε να τροποποιήσουμε την εντολή input() ώστε τον αριθμό να τον πληκτρολογήσουμε στην επόμενη γραμμή:

```
age=input("Πόσα χρόνια ζει ένας πύθωνας;\n")
```

Η προσθήκη του \n μεταφέρει την πληροφορία που θα δώσουμε από το πληκτρολόγιο στην αμέσως επόμενη γραμμή (εικόνα κάτω).

```
****************************************
Καλωσορίσατε στο πρόγραμμα 'Βρες την ηλικία μου'
****************************************


Πόσα χρόνια ζει ένας πύθωνας;
```

Στην επόμενη σελίδα θα δούμε και άλλες χρήσεις της εντολής input().

Ο αγαπημένος μας πύθωνας μπορεί να πληκτρολογήσει και κείμενο εκτός από αριθμούς!

```
name=input("Όνομα παίκτη;")
print("Καλωσόρισες",name)
```

Στην πρώτη εντολή, ζητάμε να γράψει ο παίκτης ή η παίκτρια το όνομα του/της. Το όνομα θα αποθηκευθεί στη μεταβλητή name.

Με την εντολή print, εκτυπώνεται το μήνυμα "Καλωσόρισες" και στη συνέχεια, το περιεχόμενο της μεταβλητής name (εικόνα κάτω).

```
*******************************************
Καλωσορίσατε στο πρόγραμμα 'Βρες την ηλικία μου'
*******************************************

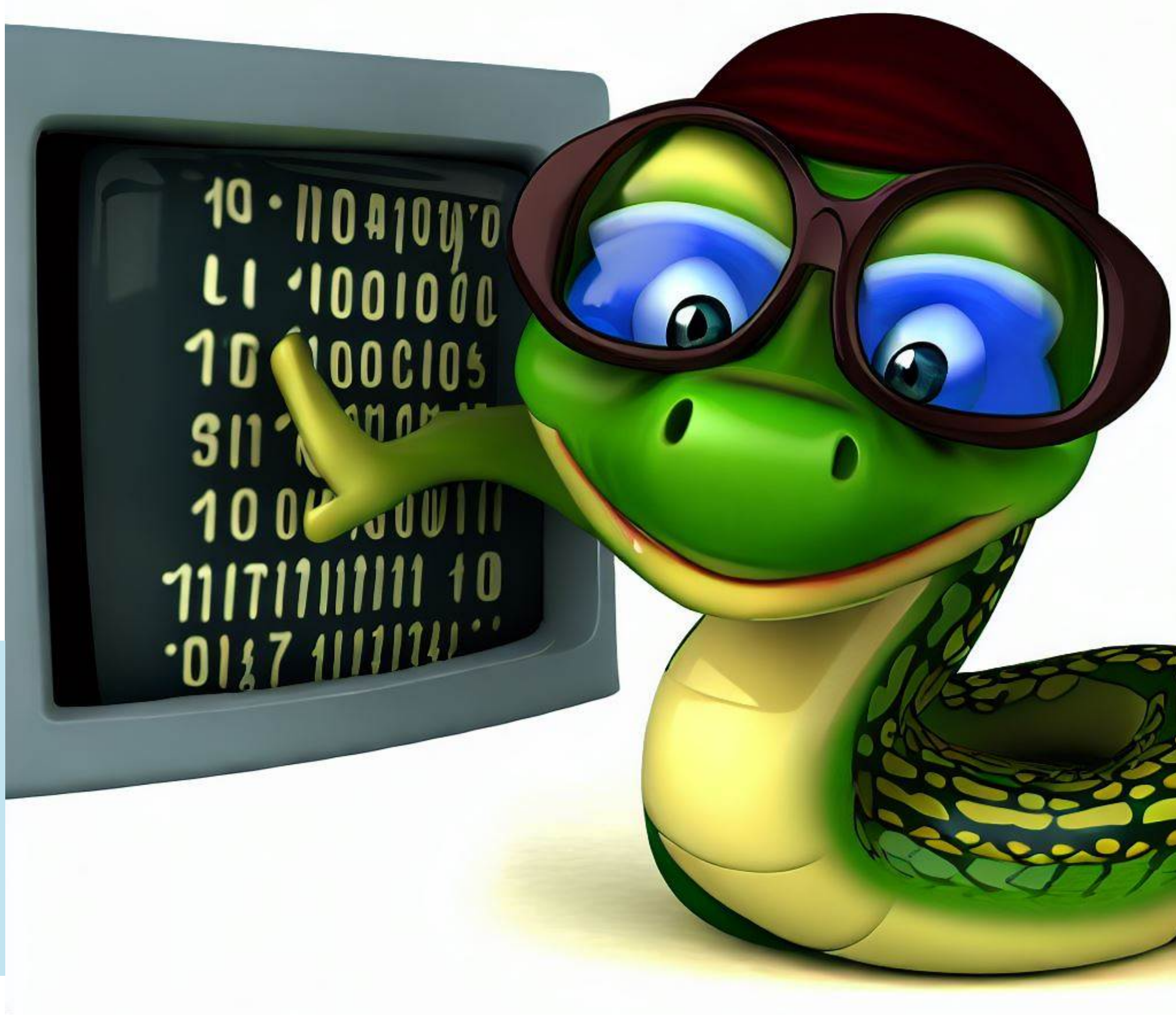

Όνομα παίκτη;Αλέξανδρος
Καλωσόρισες Αλέξανδρος
```



Λίγη γραμματική: όταν πληκτρολογούμε το όνομά μας, αυτό είναι στην ονομαστική. Όταν το προβάλλει ο υπολογιστής, αυτό συνεχίζει να παραμένει στην ονομαστική, αν και θα έπρεπε να είναι στην κλητική.

Την εντολή input() μπορούμε να τη χρησιμοποιήσουμε για να εισαγάγουμε αριθμούς με τους οποίους θα εκτελέσουμε μαθηματικές πράξεις.

Μπορούμε επίσης να χρησιμοποιήσουμε την input() για να επιλέξουμε (πολλαπλή επιλογή). Και αυτό θα το δούμε στη συνέχεια του βιβλίου μας!



Ας σχολιάσουμε...

Οι πύθωνες είναι σχολαστικά και "σχολιαστικά" ζώα. Τους αρέσει να βάζουν σημειώσεις στα κείμενά τους. Αυτό τους βοηθά να καταλαβαίνουν τι γράφει κάθε παράγραφος ή τμήμα κειμένου. Το ίδιο και όταν γράφουν προγράμματα στον υπολογιστή!

Τα σχόλια (comments), βοηθούν στο να καταλαβαίνουμε καλύτερα τι κάνει ένα πρόγραμμα ή/και μέρη ενός προγράμματος.

Για να γράψουμε σχόλιο, θα πρέπει -με κάποιο τρόπο- να πούμε στην Python πως δεν είναι εντολή για εκτέλεση, και απλά να την αγνοήσει.



myfirstpython.py - /Users/akoftero/Documents/myfirstpython.py (3.11.4)

```
print("********************************************")
print("Καλωσορίσατε στο πρόγραμμα 'Βρες την ηλικία μου'")
print("********************************************")
print()
print()
#print("Πόσα χρόνια ζει ένας πύθωνας;")
name=input("Όνομα παίκτη;")
print("Καλωσόρισες",name)
age=input("Πόσα χρόνια ζει ένας πύθωνας;\n")
```

Ln: 11 Col: 0

Στο πιο πάνω παράδειγμα, η εντολή print() εμφανίζεται με κόκκινο χρώμα. Μπροστά της έχουμε βάλει το σύμβολο "#". Το συγκεκριμένο σύμβολο, στην αρχή της εντολής, δηλώνει στην Python ότι πρέπει να την αγνοήσει στην εκτέλεση γιατί πρόκειται για σχόλιο.

Στην πιο κάτω εικόνα βλέπουμε τα σχόλια στην αρχή μιας γραμμής (εξηγούν τι κάνει ο κώδικας) αλλά και δεξιά από μια εντολή (εξηγούν τι κάνει η εντολή).

myfirstpython.py - /Users/akoftero/Documents/myfirstpython.py (3.11.4)

```
print("********************************************")
print("Καλωσορίσατε στο πρόγραμμα 'Βρες την ηλικία μου'")
print("********************************************")
print()
print()
#Με τις εντολές που ακολουθούν, ζητάμε έναν αριθμό
#για να μαντέψουμε πόσα χρόνια ζει ένας πύθωνας
name=input("Όνομα παίκτη;")
print("Καλωσόρισες",name)
age=input("Πόσα χρόνια ζει ένας πύθωνας;\n") #ζητάμε έναν αριθμό
```

Ln: 10 Col: 64

Ο αποφασιστικός πύθωνας!

Καιρός να αποφασίσει ο φιλικός πράσινος πύθωνας αν βρήκαμε σωστά την ηλικία του! Θα πρέπει -σωστά μαντέψατε- να χρησιμοποιήσουμε την εντολή “if”. Ένας πύθωνας ζει περίπου 10 χρόνια.

```
if age==10:
    print(“Βρήκατε τη σωστή ηλικία!”)
```

Στην πρώτη γραμμή, συγκρίνουμε την τιμή της μεταβλητής age (που δώσαμε από το πληκτρολόγιο) με τον αριθμό 10 (τα χρόνια που ζει ένας πύθωνας).

Στη σύγκριση, έχουμε το διπλό σύμβολο == που δηλώνει το “ίσο με”. Επίσης χρησιμοποιούμε το “μεγαλύτερο ή ίσο” (>=), “μικρότερο ή ίσο” (<=) και “μικρότερο/μεγαλύτερο”.

ΑΝ ισχύει η συνθήκη (ο αριθμός που δώσαμε είναι το 10, ΤΟΤΕ εμφανίζεται το μήνυμα “Βρήκατε τη σωστή ηλικία”.

Η εντολή print() εμφανίζεται κάτω και δεξιά της if. Όλες οι εντολές που εκτελούνται ΑΝ ισχύει η συνθήκη, πρέπει να είναι κάτω από την if και με εσοχή (indentation).

Θέλουμε, όμως, να εμφανίζει και ένα μήνυμα όταν δώσουμε διαφορετικό αριθμό. Θα πρέπει να προσθέσουμε και το ‘else’.

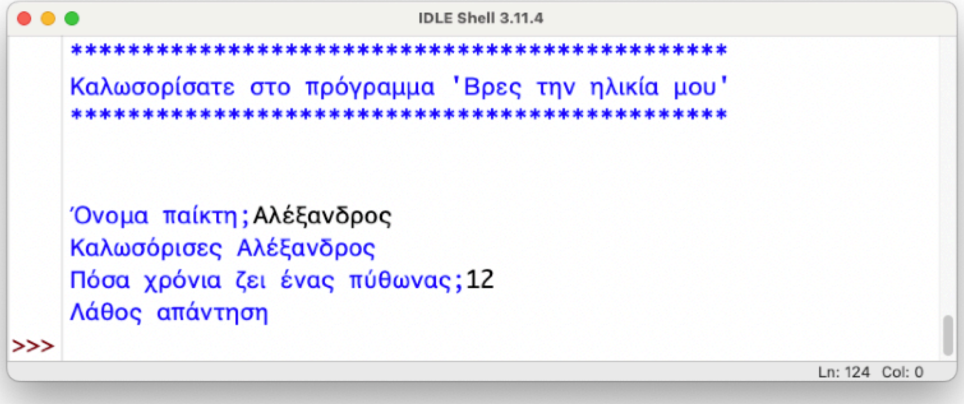
```
if age==10:
    print(“Βρήκατε τη σωστή ηλικία!”)
else:
    print(“Δυστυχώς κάνατε λάθος”)
```

ΑΝ η τιμή της μεταβλητής age είναι το 10, ΤΟΤΕ θα εμφανίσει το πρώτο μήνυμα, ΔΙΑΦΟΡΕΤΙΚΑ (else) θα εμφανίσει το δεύτερο μήνυμα.

Μπορούμε να γράψουμε τις εντολές αυτές σε μία γραμμή:

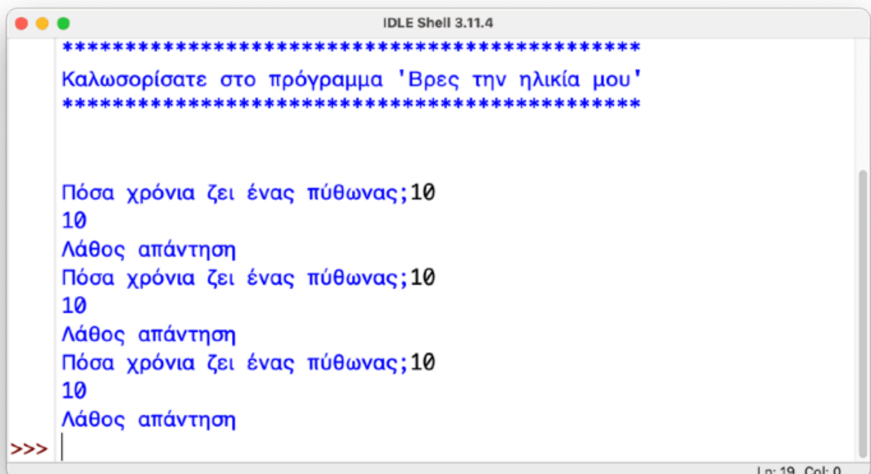
```
print(“Σωστά!”) if age==10 else print(“Λάθος”)
```

Στην εικόνα πιο κάτω, βλέπουμε το αποτέλεσμα του προγράμματος, όταν δώσουμε λάθος απάντηση. Στην επόμενη σελίδα θα μάθουμε πώς μπορούμε να συνεχίσουμε το παιχνίδι και να δώσουμε ξανά απάντηση.



Σφάλμα στον κώδικα!

Στην προηγούμενη σελίδα, είδαμε το μήνυμα που βγάζει το πρόγραμμα μας, αν δώσουμε λάθος αριθμό για την ηλικία του πύθωνα. Τι συμβαίνει όμως όταν δώσουμε τον σωστό αριθμό;



Όπως βλέπουμε και στην πιο πάνω εικόνα, παρόλο που πληκτρολογούμε τη σωστή απάντηση (που είναι το 10), το θεωρεί λάθος! Μα γιατί; αφού ο κώδικάς μας είναι σωστός. Ή κάτι πάει λάθος…;

Ακέραιοι και πορτοκάλια…

Σε προηγούμενη σελίδα, είχαμε μιλήσει για είδη μεταβλητών. Κάποιες μεταβλητές, οι τιμές που παίρνουν είναι ακέραιοι αριθμοί (integers ή int). Άλλες είναι κείμενο (που μπορεί να περιέχει ή να αποτελείται και από αριθμούς) και ονομάζονται string. Άλλες δέχονται δεκαδικούς (floating point ή float).

Αν θέλουμε να μάθουμε μια μεταβλητή τι τύπος είναι, χρησιμοποιούμε το type().

Στον κώδικα μας έχουμε τη μεταβλητή age, στην οποία δίνουμε τιμή από το πληκτρολόγιο.

```
age=input("Πόσα χρόνια ζει ένας πύθωνας;\n")
```

Στη συνέχεια, συγκρίνουμε αυτή τη μεταβλητή για να δούμε αν ισούται με τον αριθμό 10.

```
if age==10:
```

Όμως, υπάρχει κάτι που δεν προσέξαμε: όταν δίνουμε πληροφορίες στην input, αυτόματα τις τοποθετεί στη μεταβλητή όχι ως ακέραιο (int) αλλά ως κείμενο (str)!

Έτσι, ό,τι αριθμό και αν δώσουμε στην input από το πληκτρολόγιο, θα θεωρήσει πως είναι κείμενο και όχι αριθμός και άρα η σύγκριση θα είναι πάντα λάθος (σαν να του λέμε να συγκρίνει αν τα πορτοκάλια είναι ο αριθμός 10)! Στη συνέχεια θα δούμε πώς θα το διορθώσουμε!

Τύπος μεταβλητής

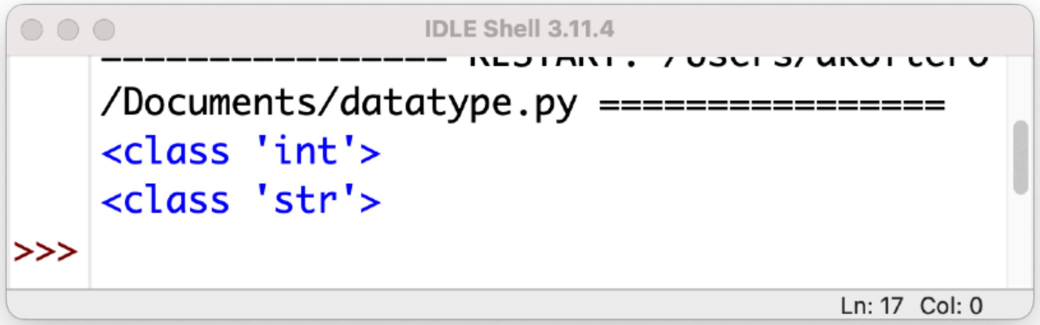
Θέλουμε να δώσουμε αριθμό από το πληκτρολόγιο, ώστε να μπορεί να τον συγκρίνει με την ηλικία του πύθωνα και να διαπιστώσει αν βρήκαμε τη σωστή απάντηση ή όχι.

Το ερώτημα είναι: πώς λέμε στην Python ότι ο αριθμός που θέλουμε να δώσουμε είναι ακέραιος και όχι κείμενο; Πρώτα από όλα, ας δούμε πώς εντοπίζουμε τον τύπο μιας μεταβλητής…

Θα χρησιμοποιήσουμε μια νέα εντολή, την type() για να ελέγξουμε τον τύπο μιας μεταβλητής. Για να εμφανίσουμε τον τύπο της μεταβλητής στην οθόνη, θα την πληκτρολογήσουμε μέσα στην print(). Για να δούμε το πιο κάτω παράδειγμα:

```
x=10      #στη μεταβλητή x δίνουμε το 10 ως αριθμό
y="10"    #στη μεταβλητή y δίνουμε το 10 ως κείμενο
print(type(x))
print(type(y))
```

Η type(x) εντοπίζει το είδος της μεταβλητής x. Όμως, για να εμφανιστεί το περιεχόμενο της στην οθόνη, πρέπει να την βάλουμε μέσα στην print(). Το ίδιο και για τη μεταβλητή y. Εκτελούμε το πρόγραμμα για να δούμε το αποτέλεσμα:



Η πρώτη μεταβλητή (x=10) αναγνωρίζεται ως ακέραιος (<class 'int'>). Η δεύτερη μεταβλητή, αν και περιέχει τον αριθμό 10, επειδή είναι σε εισαγωγικά (y="10"), αναγνωρίζεται ως κείμενο (αλφαρηθμητικό - str), γι'αυτό και στη δεύτερη γραμμή (εικόνα πάνω) εμφανίζεται το μήνυμα <class 'str'>.

Οι μεταβλητές τύπου str (string) χρησιμοποιούνται για την εισαγωγή κειμένου, αλλά μπορεί να περιλαμβάνουν και αριθμούς (ή και μόνο αριθμούς), αλλά και σύμβολα. Γι'αυτό και ονομάζεται "αλφαρηθμητική" (string).

Στη συνέχεια θα λύσουμε το πρόβλημα της εισαγωγής αριθμού ως ακεραίου από το πληκτρολόγιο!

Μεταβλητή ως Ακέραιος

Αν δοκιμάσουμε να δώσουμε έναν αριθμό από το πληκτρολόγιο, με την input(), θα αναγνωρίζεται ως κείμενο (αλφαρηθμητικό) και όχι ως ακέραιος αριθμός (σημείωση: το ίδιο ισχύει και για δεκαδικούς).

Ας δούμε την εντολή με την οποία δίνουμε πληροφορία με το πληκτρολόγιο, στη μεταβλητή age.

```
age=input(“Παρακαλώ δώστε έναν αριθμό”)
```

Θα κάνουμε μια αλλαγή στον πιο πάνω κώδικα, ώστε η τιμή της μεταβλητής age να είναι ακέραιος:

```
age=int(input(“Παρακαλώ δώστε έναναριθμό”))
```

Η πιο πάνω αλλαγή της εντολής input() λύνει το πρόβλημα, καθώς ενημερώνουμε την Python ότι θα δώσουμε αριθμό (ακέραιο) από το πληκτρολόγιο.

Προσέξτε τις διπλές παρενθέσεις στο τέλος της εντολής πιο πάνω. Αυτό οφείλεται στο ότι η input() θα πρέπει να μπει μέσα στην int().

```
int( input(“Enter number”) )
```

Ο σωστός κώδικας με την αλλαγή είναι:

myfirstpython.py - /Users/akoftero/Documents/myfirstpython.py (3.11.4)

```
print("**********************************************")
print("Καλωσορίσατε στο πρόγραμμα 'Βρες την ηλικία μου'")
print("**********************************************")
print()
print()
#Με τις εντολές που ακολουθούν, ζητάμε έναν αριθμό
#για να μαντέψουμε πόσα χρόνια ζει ένας πύθωνας

count=0 #μετρά τις προσπάθειες που κάνουμε
age=1 #αρχική τιμή του age
while count<3:
    age=int(input("Πόσα χρόνια ζει ένας πύθωνας;")) #ζητάμε έναν αριθμό
    print(age)
    if age==10:
        print("Σωστή απάντηση")
        break
    else:
        print("Λάθος απάντηση")

    count=count+1 #αυξάνει η τιμή του count κατά 1
```

Ln: 12 Col: 1

Αν δοκιμάσουμε τώρα να εκτελέσουμε το πρόγραμμα, θα δούμε πως αναγνωρίζει το 10 ως τη σωστή απάντηση!

IDLE Shell 3.11.4

```
Πόσα χρόνια ζει ένας πύθωνας;10
10
Σωστή απάντηση
>>>
```

Ln: 27 Col: 0

"Έντομα" στον κώδικα!

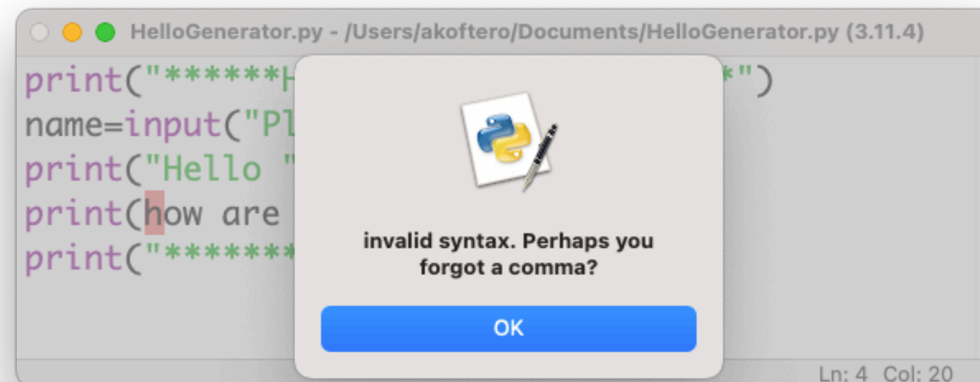
Οι πύθωνες, όταν μεταφράζουν από μια γλώσσα σε μια άλλη, προτιμούν να το κάνουν πρόταση με πρόταση. Άλλοι, προτιμούν να παίρνουν το κείμενο (ή ό,τι έχει να πει κάποιος ομιλητής) και να το μεταφράζουν ολόκληρο.

Αυτό έχει και πλεονεκτήματα και μειονεκτήματα (που δε θα αναλύσουμε σε αυτό το βιβλίο). Όμως, καθώς "τρέχει" ένα πρόγραμμα η Python, εκτελεί τις εντολές με τη σειρά.



Στην εικόνα (αριστερά) βλέπουμε πως υπάρχει λάθος στην τέταρτη γραμμή: το κείμενο δεν έχει μπει σε εισαγωγικά.

Κατά την εκτέλεση του προγράμματος, η Python θα μας ενημερώσει άμεσα για το συγκεκριμένο λάθος και θα υποδείξει τη γραμμή στην οποία βρίσκεται.



Το λάθος πιο πάνω είναι συντακτικό, επειδή λείπουν ή βάλαμε λάθος κάποιους χαρακτήρες. Στη συνέχεια θα δούμε και άλλα λάθη.

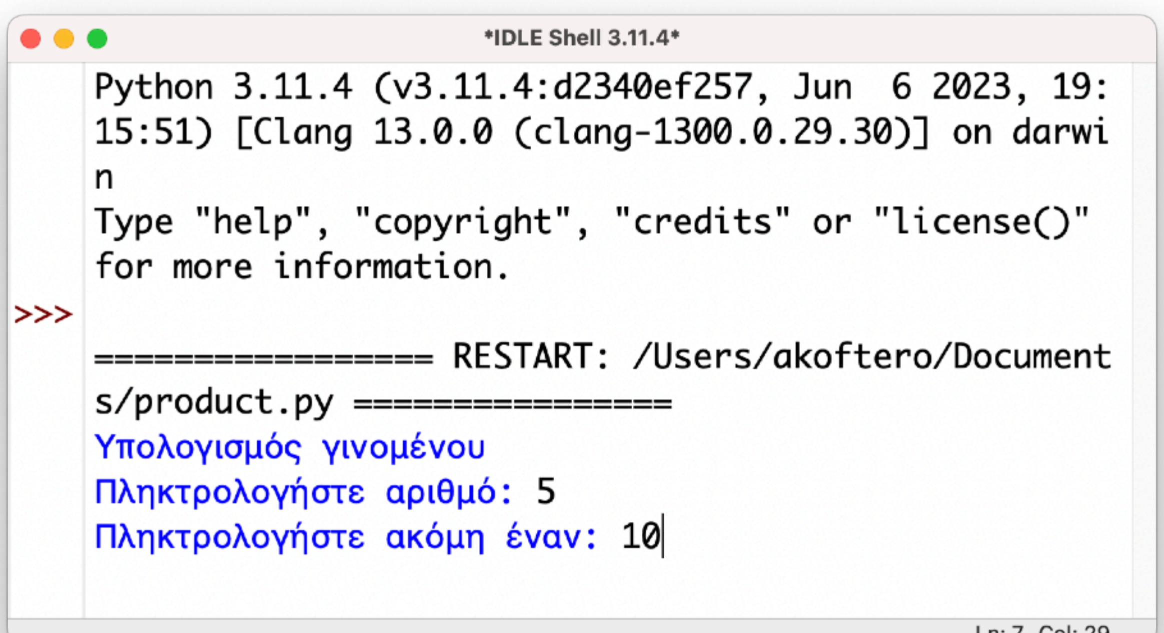
Ο όρος "bug" (έντομο) χρησιμοποιείται από το 1843 και λέγεται ότι τον δημιούργησε ο Θωμάς Έντισον, ο διάσημος εφευρέτης. Το 1947, χρησιμοποιήθηκε από την Γκρέης Χόπερ, εξαιτίας ενός σκώρου που προκάλεσε βραχυκύκλωμα στον υπολογιστή Μαρκ 2!



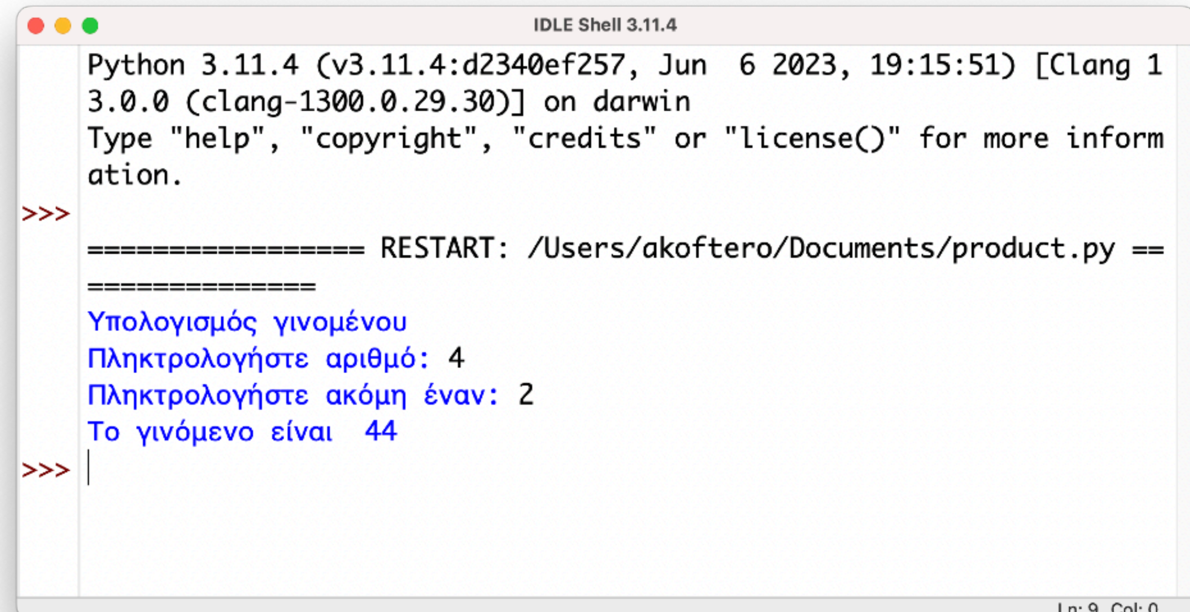
Ας δούμε ένα πιο "σοβαρό" λάθος. Θα δημιουργήσουμε ένα απλό πρόγραμμα με το οποίο υπολογίζουμε το γινόμενο δύο αριθμών:

```
print("Υπολογισμός γινομένου")
number1=input("Πληκτρολογήστε έναν αριθμό: ")
number2=int(input("Πληκτρολογήστε ακόμη έναν: "))
print("Το γινόμενο είναι ", number1*number2)
```

Δεν υπάρχει κανένα "λάθος" στον πιο πάνω κώδικα.Κατά την εκτέλεση του, όντως θα μας ζητήσει να δώσουμε 2 αριθμούς από το πληκτρολόγιο. Κάθε αριθμός θα μπει σε μια μεταβλητή (number1 και number2). Όταν τρέξουμε το πρόγραμμα, θα εκτελεστούν κανονικά οι πρώτες 3 γραμμές:



Έχουμε δώσει 2 αριθμούς από το πληκτρολόγιο. Όταν πατήσουμε το πλήκτρο ENTER για να προχωρήσει στην εκτέλεση και της τελευταίας εντολής, εμφανίζεται το ακόλουθο αποτέλεσμα:



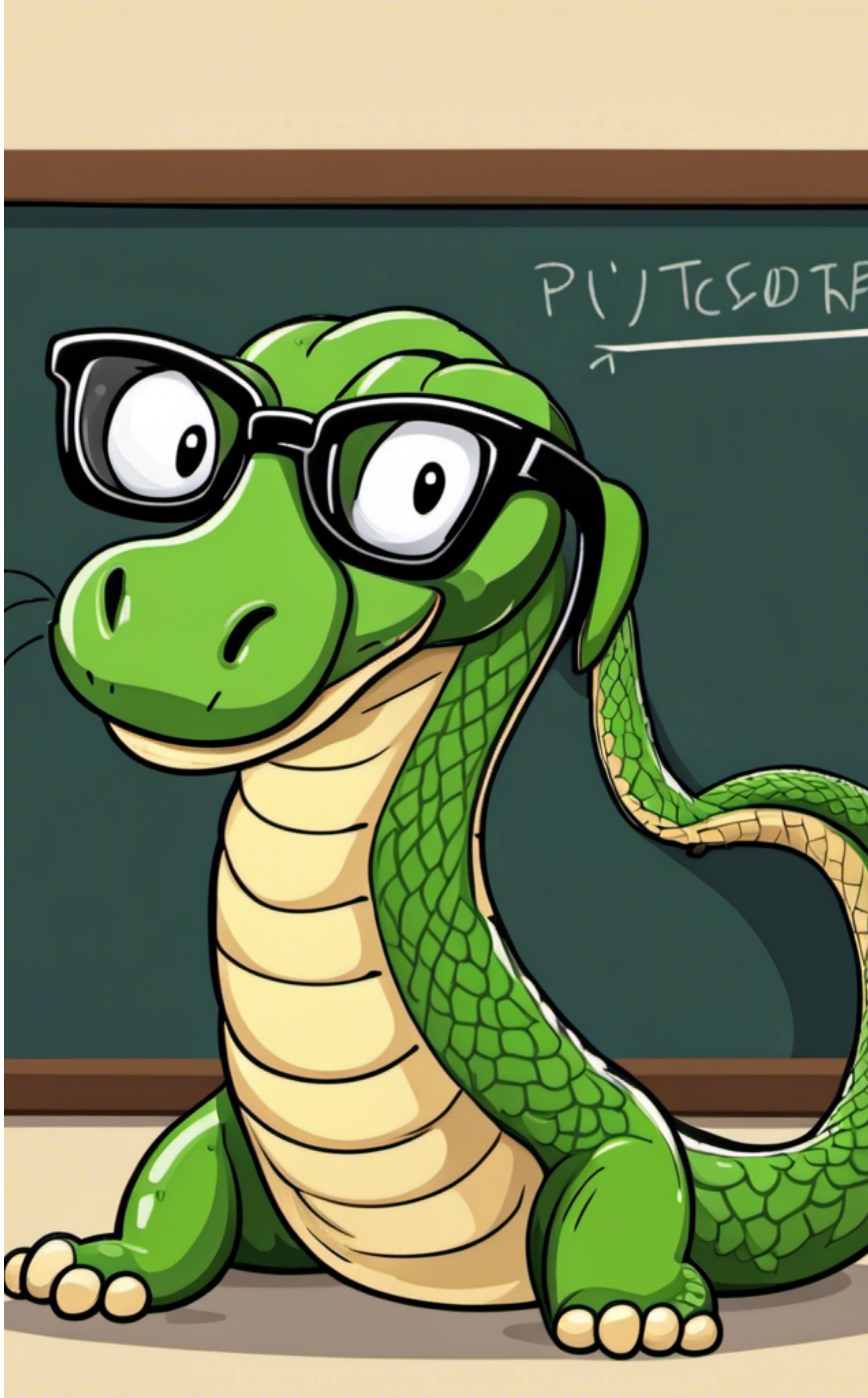
Αν το τρέξουμε ξανά, ακόμη και με τους ίδιους αριθμούς (4 και 2), θα δούμε πως βγαίνει λανθασμένο γινόμενο. Ο λόγος είναι απλός: η πρώτη μεταβλητή παίρνει μια τιμή "αλφαρηθμητική" και όχι ακέραιο αριθμό! Θα πρέπει να την αλλάξουμε σε :

```
number1=int(input("Πληκτρολογήσετε έναν αριθμό"))
```

Με αυτή την αλλαγή, ο αριθμός που θα δώσουμε στη μεταβλητή αυτή να αναγνωριστεί από τον κώδικα μας ως ακέραιος!

Τι μάθαμε μέχρι τώρα

- Στο κεφάλαιο 3 εργαστήκαμε με κώδικα πολλών γραμμών και δημιουργήσαμε ένα απλό παιχνίδι
- Με τις εντολές print() μπορούμε να εμφανίσουμε στην οθόνη, ταυτόχρονα, πολλές γραμμές με πληροφορίες
- Οι μεταβλητές χρησιμεύουν για να αποθηκεύουν τιμές (κείμενο, ακέραιος, δεκαδικός…) τις οποίες χρησιμοποιούμε στον κώδικα, συνήθως για σκοπούς σύγκρισης, μαθηματικών πράξεων κ.α.
- Με την εντολή input() μπορούμε να δώσουμε από το πληκτρολόγιο τιμή σε μια μεταβλητή
- Με την εντολή if…else μπορούμε να ελέγξουμε αν ισχύει μία συνθήκη (π.χ. 10 > 5)
- Με το type() μπορούμε να διαπιστώσουμε τον τύπο της μεταβλητής
- Με την int(input()) εισαγάγουμε μια τιμή ως ακέραιο.



Δραστηριότητες

1. Ποιο είναι το αποτέλεσμα της εκτέλεσης του πιο κάτω κώδικα; Προσπαθήστε να το γράψετε χωρίς να εκτελεστεί στον υπολογιστή.

```
name=input("Παρακαλώ γράψτε το όνομά σας")
surname=input("Παρακαλώ γράψτε το επίθετό σας")
print("Καλωσόρισες",name, surname)
```

2. Τι θα εμφανιστεί στην οθόνη όταν εκτελεστεί ο πιο κάτω κώδικας; Να το γράψετε χωρίς να εκτελεστεί στον υπολογιστή.

```
print("*************************")
print("*   Υπολογισμός δυνάμεων  *")
print("*************************")
print()
base=int(input("Παρακαλώ δώστε έναν αριθμό"))
power=int(input("Παρακαλώ δώστε τη δύναμη"))
print("Το",base,"στη δύναμη",power,"ισούται
με",base**power)
```

3. Να γράψετε ένα πρόγραμμα το οποίο να ζητά από το πληκτρολόγιο έναν αριθμό από το 1-10. Στη συνέχεια, να εμφανίζει τον πίνακα του πολλαπλασιασμού αυτού του αριθμού.

4. Να γράψετε ένα πρόγραμμα με το οποίο να ζητάτε τον "κωδικό" από το πληκτρολόγιο. Αν ο κωδικός είναι σωστός ("mypassword"), να εμφανίζεται το μήνυμα "Πολύ σωστά". Διαφορετικά να σας βγάζει το μήνυμα "κάνατε λάθος" και να σταματά.

5. Να διορθώσετε τον πιο κάτω κώδικα ώστε να λειτουργεί σωστά το πρόγραμμα.

```
@Υπολογισμός γινόμενου δύο αριθμών
print(Καλωσορίσατε στο πρόγραμμά μας)
number1=input("Δώστε τον πρώτο αριθμό")
number2=input("Δώστε τον δεύτερο αριθμό")
print("Το γινόμενό τους
είναι",number1*Number2)
```

4. Επαναλήψεις…

```
10 |  print (“It takes all the running you can do,
20 | to keep in the same place. If you want to get
30 | somewhere else, you must run at least
40 | twice as fast as that!”)
50 | #Alice in Wonderland
```

Επαναλήψεις

Όταν γράφουμε προγράμματα, πολλές φορές χρειάζεται να επαναλάβουμε κάποιες εντολές. Για παράδειγμα, στο παιχνίδι “Μάντεψε τον αριθμό που σκέφτομαι”, θα πρέπει να δώσουμε 2 ή και 3 ευκαιρίες σε κάποιον να το βρει. Ένας τρόπος είναι να γράψουμε όλες τις εντολές 3 φορές. Άλλος τρόπος είναι, με κάποιες εντολές, να επαναλάβουμε 3 (ή και περισσότερες) φορές την εκτέλεση τους.

Οι επαναλήψεις (loops) είναι ιδιαίτερα χρήσιμες γιατί μας επιτρέπουν να δημιουργήσουμε πολύπλοκα προγράμματα χωρίς να χρειάζεται να επαναλαμβάνουμε συνέχεια τις ίδιες εντολές (σε κάποιες περιπτώσεις θα χρειαστεί και αυτό).



Τι θα γνωρίσουμε:

Στο **Κεφάλαιο 4 "Επαναλήψεις..."** θα γνωρίσουμε τα πλεονεκτήματα της χρήσης loops για εκτέλεση εντολών που επαναλαμβάνονται. Επίσης θα:

- Μάθουμε την while() και θα τη χρησιμοποιήσουμε σε συνδυασμό με την if()
- Χρησιμοποιήσουμε μεταβλητές για να ελέγξουμε τον αριθμό επαναλήψεων
- Μάθουμε και θα χρησιμοποιήσουμε την επανάληψη for σε συνδυασμό με την range()
- Εκτελέσουμε επαναλήψεις με συγκεκριμένο εύρος αριθμών
- Εκτελέσουμε επαναλήψεις για δημιουργία σχημάτων και για υπολογισμό γινόμενου αριθμών



Ο επαναληπτικός Πύθωνας!

Οι πύθωνες είναι ιδιαίτερα πεισματάρικα ζώα - μπορούν να κάνουν κάτι ξανά και ξανά, μέχρι να γίνει σωστά ή μέχρι να βαρεθούν - ό,τι έρθει πρώτο!

Στην προηγούμενη ενότητα, δημιουργήσαμε ένα απλό παιχνίδι για την ηλικία του πύθωνα. Το παιχνίδι μας, αν δώσουμε λάθος απάντηση, θα τελειώσει! Εμείς όμως θέλουμε να μπορούμε να συνεχίσουμε μέχρι να δώσουμε τη σωστή απάντηση (ή, έστω, να δώσουμε 2 ή 3 απαντήσεις πριν τελειώσει).

Οι εντολές αυτές ονομάζονται “επαναλήψεις” (loops). Υπάρχουν διαφορετικοί τύποι εντολών επανάληψης που μπορούμε να χρησιμοποιήσουμε. Για το παιχνίδι μας, θα χρησιμοποιήσουμε την εντολή while.



Αυτό που θέλουμε είναι να συνεχίζει το παιχνίδι, μέχρι να δώσουμε από το πληκτρολόγιο τον αριθμό 10. Ας δούμε την εντολή:

```
    while age!=10:
```

Στην πιο πάνω εντολή, όσο η τιμή που πληκτρολογούμε είναι διαφορετική από το 10, θα συνεχίσει το πρόγραμμα να εκτελεί τις εντολές:

```
age=input(“Πόσα χρόνια ζει ένας πύθωνας;\n”)
```

Η εντολή αυτή είναι απαραίτητη, ώστε να συνεχίσει να μας ζητά να δώσουμε άλλον αριθμό.

```
if age==10:
    print(“Σωστά!”)
else:
    print(“Λάθος”)
```

Σημαντικό: πριν την εντολή while, θα πρέπει να δώσουμε μια αρχική τιμή στη μεταβλητή που θα χρησιμοποιήσουμε. Η τιμή αυτή δε θα πρέπει να είναι η ίδια με αυτήν που ελέγχει (π.χ. οτιδήποτε εκτός από 10).



Στη συνέχεια θα μελετήσουμε και άλλα παραδείγματα για τις επαναλήψεις.

Στο προηγούμενο παράδειγμα, είδαμε τον τρόπο με τον οποίο δίνουμε ξανά και ξανά την ηλικία του Πύθωνα, μέχρι να βρούμε το σωστό αποτέλεσμα. Αυτό μπορεί να πάρει ώρες (ή και μέρες!) αν δεν βάλουμε κάποιο περιορισμό.

Θα αλλάξουμε λίγο τον κώδικα μας, ώστε να σταματά να ζητά αριθμό μετά από 3 προσπάθειες!

Ξεκινάμε με μια νέα μεταβλητή, την οποία θα χρησιμοποιήσουμε ως "μετρητή":

```
count=1
```

Με την πιο πάνω μεταβλητή, θα μετράμε τις προσπάθειές μας (και άρα, τόσες επαναλήψεις θα κάνουμε).

```
while count<3:
```

Με την πιο πάνω αλλαγή στο while, θα συνεχίζει η επανάληψη για όσο η τιμή της μεταβλητής count είναι μικρότερη από 3.

```
age=input("Πόσα χρόνια ζει ένας πύθωνας;\n")
print("Σωστά!") if age==10 else print("Λάθος")
```

Με τις πιο πάνω εντολές, παίρνουμε μια τιμή από το πληκτρολόγιο, η οποία θα αποθηκευτεί στη μεταβλητή age. Στην επόμενη γραμμή, γίνεται έλεγχος κατά πόσο το age έχει την τιμή 10.

Εδώ θα πρέπει να προσθέσουμε ακόμη μια γραμμή: μετά τον έλεγχο με το if, θα πρέπει να αυξήσουμε την τιμή του count κατά 1. Αυτό είναι σημαντικό, γιατί είπαμε πως η επανάληψη θα γίνει μόνο 3 φορες. Ο κώδικας είναι:

```
count=count+1
```

Στον προγραμματισμό, οι εντολές διαβάζονται από δεξιά προς αριστερά. Η πιο πάνω εντολή διαβάζεται ως "πρόσθεσε 1 στην τιμή που έχει ήδη η count, και αποθήκευσε το νέο περιεχόμενο στην ίδια μεταβλητή (count).



Την αλλαγή στην τιμή του count μπορούμε να τη γράψουμε και διαφορετικά:

```
count+=1
```

Φαίνεται λίγο περίεργο, όμως το διαβάζουμε ως εξής: "στην τιμή του count, θα πρέπει να προσθέσεις 1".

Μόνο που εδώ έχουμε ένα πρόβλημα: αν τρέξουμε το πρόγραμμα μας, τερματίζεται στις δύο προσπάθειες!

Στην προηγούμενη σελίδα, είδαμε την επανάληψη των εντολών μας. Όμως, αντί για 3 επαναλήψεις, έγιναν μόνο 2! Για να λύσουμε αυτό το "μυστήριο".

```
count=1
while count<3:
      count=count+1
```

Αρχικά η count έχει την τιμή 1

Το while ελέγχει κατά πόσο η count είναι μικρότερη από 3. Επειδή ισχύει αυτό (count=1), προχωρά με την επόμενη εντολή (count=count+1). Τώρα η count είναι 2. Αυτή είναι η **πρώτη επανάληψη**.

Το while ελέγχει ξανά κατά πόσο το count είναι μικρότερο από 3. Όπως είδαμε, το count=2, άρα συνεχίζει την εκτέλεση των εντολών. Η επόμενη εντολή λέει στο count να αυξήσει την τιμή του κατά 1. Αφού το count είχε την τιμή 2, τώρα θα γίνει 3 (count=2+1). Αυτή είναι η **δεύτερη επανάληψη**.

Το count έχει την τιμή 3 τώρα. Στον έλεγχο του while (κατά πόσο το count είναι μικρότερο του 3), βλέπουμε ότι πλέον δεν ισχύει! Και σταματά να ισχύει στη δεύτερη επανάληψη!

Για να λειτουργήσει το πρόγραμμα μας όπως το θέλουμε (να μας δώσει τουλάχιστο 3 προσπάθειες) θα πρέπει να κάνουμε την ακόλουθη αλλαγή:

```
count=0 (άρα ξεκινά από το 0 και όχι το 1)
while count<3:
    count=count+1
```

Αυτή είναι μια λύση. Άλλη λύση είναι να κρατήσουμε το 1 ως αρχική τιμή του count και απλά να αυξήσουμε την τιμή στον έλεγχο.

```
count=1
while count<4:
    count=count+1
```

Μόνο που… υπάρχει ακόμη ένα πρόβλημα με τον κώδικά μας (ίσως να το έχετε μαντέψει ήδη!).

Στην επόμενη σελίδα θα κάνουμε ακόμη μια αλλαγή στον κώδικα, ώστε το πρόγραμμα να δουλεύει σωστά.



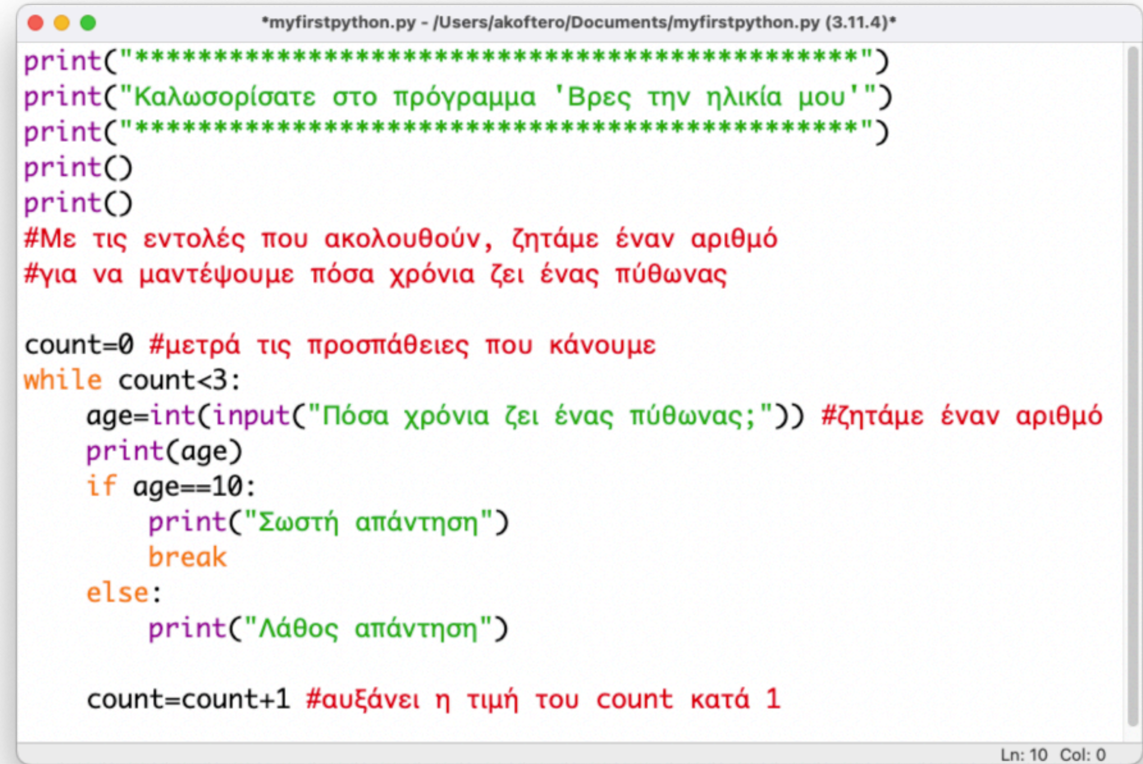
Breaking Good!

Στον πιο πάνω κώδικα, η επανάληψη συνεχίζεται για 3 προσπάθειες. Όμως, ακόμη και αν βρούμε τη σωστή ηλικία από την πρώτη φορά, θα συνεχίσει για άλλες 2!

Αυτό διορθώνεται εύκολα με την εντολή break και κάποιες αλλαγές στη συνθήκη if.

```
count=1 #αρχική τιμή της count
while count<4: #ξεκινά η επανάληψη
    age=int(input(“Πόσα χρόνια ζει ένας πύθωνας;”)
    if age==10:
        print(“Σωστή απάντηση”)
        break #σταματά η εκτέλεση της while
    else:
        print(“Λάθος απάντηση”)
    count=count+1 #τελειώνει η επανάληψη
```




myfirstpython.py - /Users/akoftero/Documents/myfirstpython.py (3.11.4)

```
print("************************************************")
print("Καλωσορίσατε στο πρόγραμμα 'Βρες την ηλικία μου'")
print("************************************************")
print()
print()
#Με τις εντολές που ακολουθούν, ζητάμε έναν αριθμό
#για να μαντέψουμε πόσα χρόνια ζει ένας πύθωνας

count=0 #μετρά τις προσπάθειες που κάνουμε
while count<3:
    age=int(input("Πόσα χρόνια ζει ένας πύθωνας;")) #ζητάμε έναν αριθμό
    print(age)
    if age==10:
        print("Σωστή απάντηση")
        break
    else:
        print("Λάθος απάντηση")

    count=count+1 #αυξάνει η τιμή του count κατά 1
```

Ln: 10 Col: 0

Στην εικόνα εμφανίζεται ολόκληρος ο κώδικας που έχουμε γράψει! Έχουμε δημιουργήσει ένα απλό παιχνίδι, με 18 γραμμές κώδικα, στο οποίο χρησιμοποιήσαμε μεταβλητές, επανάληψη (while), σχόλια καθώς και συνθήκη ελέγχου της τιμής της μεταβλητής (If…else).

Στην εικόνα πάνω, βλέπουμε πως στις σειρές 8 και 19 έχουμε αφήσει κενό. Τα κενά αυτά τα αγνοεί η Python κατά την εκτέλεση των εντολών. Είναι χρήσιμα για να είναι ευανάγνωστος ο κώδικας.

One, Two, Three, For!

Για επαναλήψεις, μπορούμε να χρησιμοποιήσουμε και την εντολή For.

Θα δημιουργήσουμε ένα νέο παιχνίδι, στο οποίο θα πρέπει να μαντέψουμε έναν αριθμό που “σκέφτεται” ο υπολογιστής! Θα χρησιμοποιήσουμε την επανάληψη For, καθώς και τη συνάρτηση range().

Η range() είναι μια σημαντική συνάρτηση. Μας βοηθά να χρησιμοποιήσουμε ένα εύρος τιμών (από έναν αριθμό σε έναν άλλον).

Ας δούμε τις εντολές:

```
for counter in range(3):
    print(counter)
```

Ας εξετάσουμε τις εντολές:

Το counter είναι μια μεταβλητή. Με τη συνάρτηση range(3), η μεταβλητή παίρνει τιμές από το 0 (που είναι η αρχική τιμή) μέχρι και το 2. Ο αριθμός 3 μας λέει, δηλαδή, να πάρουμε 3 αριθμούς με τη σειρά, ΑΛΛΑ ο πρώτος είναι το 0 (άρα οι αριθμοί είναι 0, 1 και 2).

Η εντολή for θα εκτελέσει το πρόγραμμα 3 φορές. Στην πρώτη εκτέλεση, θα τυπώσει στην οθόνη τον αριθμό 0. Στη δεύτερη εκτέλεση, τον αριθμό 1 και στην τρίτη εκτέλεση, τον αριθμό 2.

```
0
1
2
```

Ας δούμε ένα παράδειγμα χρήσης του for για προβολή πίνακα πολλαπλασιασμού:

```
for counter in range(1, 13):
```

Στη γραμμή πιο πάνω, θέτουμε το εύρος να είναι από το 1 μέχρι το 13. Αν βάζαμε range(13), τότε θα ξεκινούσε από το 0 μέχρι το 12. Εμείς θέλουμε να ξεκινά από το 1 μέχρι το 12. Προσθέτουμε και την επόμενη εντολή:

```
print(8*counter) #θα εμφανιστεί ο πίνακας του 8
```



While... or...

Στη συνέχεια θα γνωρίσουμε πώς κάνουμε επανάληψη μέσα σε μια άλλη επανάληψη. Αυτό ονομάζεται "nested loop" ή "φώλιασμα" επανάληψης.

Ας δούμε τη χρήση της while() για τη δημιουργία ενός προγράμματος για τον υπολογισμό όλων των πινάκων πολλαπλασιασμού, από το 1-12:

```
print("******************************")
print("* Πίνακες Πολλαπλασιασμού *")
print("******************************")
print()
```

Με τις πιο πάνω εντολές δίνουμε στοιχεία στην οθόνη για το τι θα κάνει το πρόγραμμα. Στη συνέχεια θα ζητήσουμε να επιλέξει ο/η χρήστης τον πίνακα του πολλαπλασιασμού:

```
pinakas=int(input("Επιλέξτε πίνακα από το 1-12"))
```

Θέλουμε να περιορίσουμε την επιλογή από το 1-12. Θα χρησιμοποιήσουμε το while() για να ελέγξουμε αν ο αριθμός που μας δίνουν από το πληκτρολόγιο είναι από το 1-12. Θέλουμε να γίνεται διπλός έλεγχος: αν μας δώσουν αριθμό μικρότερο του 1 (π.χ. 1) να βγάζει μήνυμα ότι πρέπει να δώσουμε μεγαλύτερο. Αν μας δώσουν αριθμό μεγαλύτερο από το 12, τότε να βγάζει μήνυμα ότι πρέπει να δώσουμε μικρότερο αριθμό.

Αν χρησιμοποιούσαμε το "if" από μόνο του, θα έκανε τον έλεγχο, όμως αυτός θα γινόταν μόνο μια φορά (χωρίς επανάληψη). Ας δούμε ένα παράδειγμα με το while:

```
while pinakas<1:
```

η πιο πάνω γραμμή ελέγχει συνεχώς αν η τιμή που δίνουμε στη μεταβλητή "pinakas" είναι μικρότερη από το 1. Αν βρει ότι η τιμή που δίνουμε είναι μικρότερη του 1, τότε εμφανίζει την πιο κάτω εντολή:

```
pinakas=int(input("Παρακαλώ δώστε αριθμό ίσο
ή μεγαλύτερο του 1"))
```

Η πιο πάνω είναι πολύ σημαντική, γιατί θα την εμφανίζει συνέχεια μέχρι να δώσουμε αριθμό μεγαλύτερο από το 1! Επίσης, εμφανίζει και το μήνυμα για το τι αριθμό περιμένει να του δώσουμε!

Είναι πολύ σημαντικό να δίνουμε πληροφορίες για το τι πληροφορία θέλουμε από το πληκτρολόγιο. Έτσι, είναι σημαντικό να ενημερώσουμε (πιο πάνω) πως χρειάζεται να δώσουμε αριθμό μεγαλύτερο από το 1 για να συνεχίσει.



Οι εντολές που δώσαμε μέχρι τώρα, μας ενημερώνουν για το τι κάνει το πρόγραμμα (Πίνακες πολλαπλασιασμού) και μας ζητά να επιλέξουμε τον πίνακα που θέλουμε και τον αποθηκεύει στη μεταβλητή "pinakas" ως ακέραιο (int). Στη συνέχεια, ελέγχει αν ο αριθμός που δώσαμε είναι μικρότερος του 1 (επειδή θέλουμε να επιλέξει από τον πίνακα του 1 μέχρι του 12).

Όμως, θα πρέπει να ελέγξει και αν ο αριθμός που δίνουμε είναι μεγαλύτερος από το 12! Αν πληκτρολογήσουμε, για παράδειγμα, το 14, θα πρέπει να μας ενημερώνει ότι ο αριθμός πρέπει να είναι ίσος ή μικρότερος του 12 και να ζητά να δώσουμε έναν άλλο αριθμό.

```
while pinakas>12:
    pinakas=int(input("Παρακαλώ δώστε αριθμό 
ίσο ή μικρότερο του 12”))
```

Με την πιο πάνω εντολή, το πρόγραμμά μας ελέγχει συνεχώς αν ο αριθμός που πληκτρολογήσαμε είναι μεγαλύτερος του 12. Αν είναι μεγαλύτερος, τότε μας εμφανίζει μήνυμα για να δώσουμε αριθμό ίσο ή μικρότερο του 12.

Όμως, υπάρχει πρόβλημα… Αν εκτελέσουμε τον κώδικα, θα δούμε πως ο έλεγχος γίνεται, πράγματι, όμως γίνεται μόνο την πρώτη φορά! Και δεν ελέγχει ταυτόχρονα την τιμή που έχουμε δώσει. Εδώ θα πρέπει να κάνουμε μια αλλαγή στον κώδικα: να χρησιμοποιήσουμε την while, όμως να του πούμε να ελέγχει κατά πόσο ο αριθμός είναι μικρότερος του 1 ή μεγαλύτερος του 12.

```
while pinakas<1 or pinakas>12:
```

Με την πιο πάνω εντολή, επαναλαμβάνει όλες όσες περιλαμβάνονται στην επανάληψη (loop) αν ο αριθμός είναι μικρότερος του 1 ή μεγαλύτερος του 12. Θέλουμε όμως να μας βγάζει μήνυμα για να ξέρουμε τι λάθος κάναμε και τι αριθμό να δώσουμε. Έτσι έχουμε τις εξής συνθήκες:

```
if pinakas<1:
    pinakas=int(input(“Παρακαλώ δώστε αριθμό
    ίσο ή μεγαλύτερο του 1”))
```

Με τις πιο πάνω εντολές, αν ο αριθμός που πληκτρολογήσαμε είναι μικρότερος του 1, τότε θα μας βγάλει μήνυμα να δώσουμε αριθμό ίσο ή μεγαλύτερο από το 1. Στη συνέχεια θα ελέγξουμε αν ο αριθμός που πληκτρολογήσαμε είναι μεγαλύτερος του 12.

```
else:
    pinakas=int(input(“Παρακαλώ δώστε αριθμό
    ίσο ή μικρότερο του 12”))
```

Μέχρι τώρα, ο κώδικας μας ζητά έναν αριθμό από το πληκτρολόγιο και ελέγχει αν είναι από το 1 ως το 12. Στη συνέχεια, θα εμφανίσουμε τον πίνακα πολλαπλασιασμού του αριθμού που επιλέξαμε. Θα χρησιμοποιήσουμε μια επανάληψη for:

```
for multiplier in range(1,13):
    print(pinakas,”X”,multiplier,”
=“,pinakas*multiplier)
```

```
IDLE Shell 3.11.4
==========
******************************
*    Πίνακες Πολλαπλασιασμού    *
******************************

Παρακαλώ επιλέξτε έναν πίνακα από το 1-12: 2
2 X 1 = 2
2 X 2 = 4
2 X 3 = 6
2 X 4 = 8
2 X 5 = 10
2 X 6 = 12
2 X 7 = 14
2 X 8 = 16
2 X 9 = 18
2 X 10 = 20
2 X 11 = 22
2 X 12 = 24
Ln: 111  Col: 0
```

Στην εικόνα πιο πάνω φαίνεται το αποτέλεσμα της εκτέλεσης του κώδικα μας. Έχουμε δημιουργήσει ένα αρκετά σύνθετο πρόγραμμα, με επανάληψη while, επανάληψη for καθώς και συνθήκες με το if. Ο ολοκληρωμένος μας κώδικας εμφανίζεται πιο κάτω:

```
πινακεσ.py - /Users/akoftero/Documents/πινακεσ.py (3.11.4)
print("******************************")
print("*    Πίνακες Πολλαπλασιασμού    *")
print("******************************")
print()
pinakas=int(input("Παρακαλώ επιλέξτε έναν πίνακα από το 1-12: "))

while pinakas<1 or pinakas>12:
    if pinakas<1:
        pinakas=int(input("Παρακαλώ δώστε αριθμό μεγαλύτερο του 1"))
    else:
        pinakas=int(input("Παρακαλώ δώστε αριθμό μικρότερο του 12"))
for multiplier in range(1,13):
    print(pinakas,"X",multiplier,"=",pinakas*multiplier)
Ln: 13  Col: 37
```



Το πρόγραμμα μας, αν και αρκετά σύνθετο, δεν έχει μια βασική λειτουργία: για να υπολογίσουμε και άλλον πίνακα, θα πρέπει να το τρέξουμε από την αρχή. Στη συνέχεια, θα δούμε πώς τροποποιούμε τον κώδικα ώστε να συνεχίζει με νέο πίνακα πολλαπλασιασμού, εκτός και αν διακόψουμε εμείς τη λειτουργία του! Για να γίνει αυτό θα πρέπει να τοποθετήσουμε επανάληψη μέσα στην επανάληψη. Αυτό ονομάζεται "φωλιασμένη" επανάληψη ή nested loop!

“Φωλιασμένη” Επανάληψη

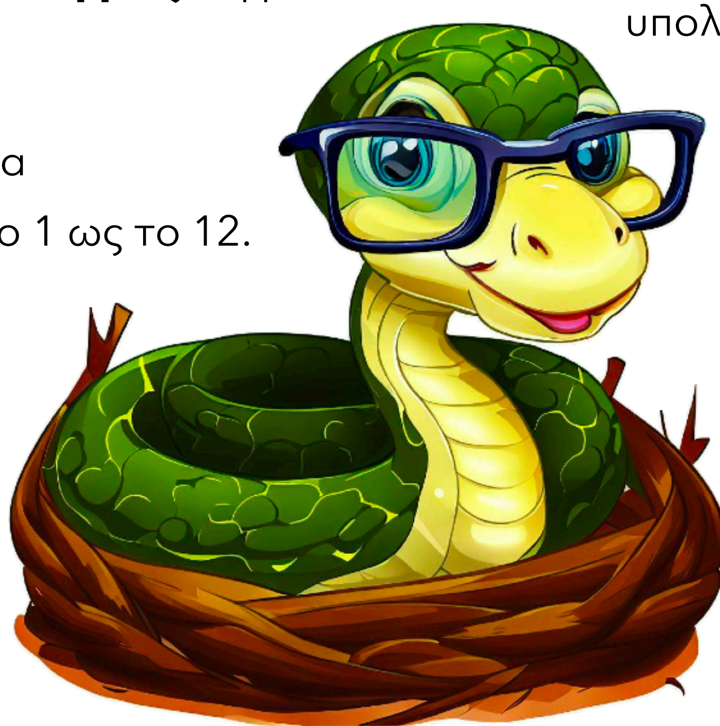
Στη συνέχεια θα γνωρίσουμε πώς κάνουμε επανάληψη μέσα σε μια άλλη επανάληψη. Όπως αναφέραμε και στην προηγούμενη σελίδα, αυτό ονομάζεται “nested loop” ή “φώλιασμα” επανάληψης.

Στο προηγούμενο πρόγραμμα, ζητούσαμε μια τιμή από το πληκτρολόγιο και στη συνέχεια εμφανιζόταν ο πίνακας πολλαπλασιασμού. Όμως, με την εμφάνιση του πίνακα, το πρόγραμμα σταματούσε.

Στη συνέχεια θα δημιουργήσουμε ένα πρόγραμμα το οποίο υπολογίζει τον πίνακα ενός αριθμού από το 1 ως το 12, και στη συνέχεια θα μας ρωτά αν θέλουμε να συνεχίσει με άλλο αριθμό.

Θα πρέπει δηλαδή να έχουμε **δύο επαναλήψεις**, τη μία μέσα στην άλλη:

- η μία επανάληψη θα υπολογίζει όλα τα πολλαπλάσια του αριθμού μας, από το 1 ως το 12.
- Η επανάληψη αυτή θα είναι μέσα σε μια άλλη επανάληψη που θα αρχίζει ξανά τη διαδικασία, με νέο αριθμό.



Για το παράδειγμα μας, θα κάνουμε ένα πιο απλό πρόγραμμα σε σχέση με το προηγούμενο.

```
print("*****************************")
print("* Πίνακες Πολλαπλασιασμού *")
print("*****************************")
print()
```

```
x=“y” #δίνουμε αρχική τιμή στη μεταβλητή x
```

Η μεταβλητή x είναι απαραίτητη, γιατί με αυτήν θα ελέγχουμε αν θα συνεχίσει ή όχι η εκτέλεση του προγράμματος.

```
while x ==“y”: #ελέγχει αν το x έχει ως τιμή το y
```

Με την πιο πάνω εντολή ξεκινά η πρώτη επανάληψη. Θα επαναλαμβάνει όλες τις εντολές που ακολουθούν, μέχρι να δώσουμε από το πληκτρολόγιο το γράμμα ‘y’.

Στη συνέχεια, δίνουμε τις υπόλοιπες εντολές για υπολογισμό του πίνακα πολλαπλασιασμού. Μέσα στην while θα έχουμε μια επανάληψη for.

```
pinakas=int(input(“Δώστε έναν αριθμό: “))
for multiplier in range(1,13)
    print(multiplier*pinakas)
```

Μόλις ολοκληρωθεί η επανάληψη for, εμφανίζεται μήνυμα για συνέχεια:

```
x=input("Πατήστε y και ENTER για συνέχεια,
οποιοδήποτε άλλο πλήκτρο για τερματισμό")
```

Με την πιο πάνω εντολή, ξαναρχίζει η επανάληψη αν πληκτρολογήσουμε το "y" (και πατήσουμε ENTER) ή τερματίζεται το πρόγραμμα. Τελευταία εντολή που μένει, είναι η print(), με την οποία θα εμφανιστεί ένα μήνυμα όταν τερματίσουμε το πρόγραμμα.

```
print("Σας ευχαριστούμε!")
```

Όλος ο κώδικας εμφανίζεται πιο κάτω:

```
print("******************************")
print("* Πίνακες Πολλαπλασιασμού *")
print("******************************")
print()

x="y" #δίνουμε αρχική τιμή στη μεταβλητή x

while x =="y": #ελέγχει αν πληκτρολογήσαμε y
        pinakas=int(input("Δώστε έναν
        αριθμό: "))
        for multiplier in range(1,13)
            print(multiplier*pinakas)
        x=input("Πατήστε y και ENTER για
        συνέχεια, οποιοδήποτε άλλο πλήκτρο για
        τερματισμό")
print("Σας ευχαριστούμε!")
```



Με τις εντολές αυτές, το πρόγραμμα θα εκτελείται συνεχώς και θα υπολογίζει τον πίνακα πολλαπλασιασμού του αριθμού που δώσαμε. Με το πάτημα του πλήκτρου "y" (και του ENTER), θα αρχίζει η διαδικασία από την αρχή, μέχρι να πατήσουμε οποιδήποτε άλλο πλήκτρο εκτός από το "y".



Το πρόγραμμα μας έχει μια μικρή δυσκολία: αν πατήσουμε το πλήκτρο "y" και είμαστε σε κεφαλαία γράμματα (π.χ. έμεινε πατημένο το Caps Lock) τότε δε θα αναγνωριστεί από τη συνθήκη (το "y" θεωρείται διαφορετικό από το "Y"). Το ίδιο θα συμβεί και αν το πληκτρολόγιο μας είναι στην ελληνική αντί στην αγγλική - θα εμφανίζεται το "υ" αντί του "y".

Στις **Δραστηριότητες**, στο τέλος του κεφαλαίου αυτού, υπάρχει σχετική άσκηση για να λύσετε αυτό το πρόβλημα!

Τι μάθαμε μέχρι τώρα

- Στο κεφάλαιο 4 γνωρίσαμε τις επαναλήψεις while και for
- Με την επανάληψη while, επαναλαμβάνεται μια σειρά εντολών για όσο ισχύει μια σχέση (π.χ. x=10)
- Με την επανάληψη for, ελέγχουμε κατά πόσο έχει γίνει συγκεκριμένος αριθμός επαναλήψεων
- Η range(0,12) μας δίνει ένα ορισμένο εύρος αριθμών.
- Σε μια επανάληψη (loop) μπορούμε να έχουμε και άλλη επανάληψη, στο εσωτερικό της πρώτης. Αυτό ονομάζεται “φώλιασμα” ή nested loop.



Δραστηριότητες

1. Ποιο είναι το αποτέλεσμα της εκτέλεσης του πιο κάτω κώδικα; Προσπαθήστε να το γράψετε χωρίς να εκτελεστεί στον υπολογιστή.

```
x=int(input("Παρακαλώ δώστε έναν αριθμό"))
for count in range(0,12)
print("Ο πίνακας του ",x,"είναι",x*count)
```

2. Τι θα εμφανιστεί στην οθόνη όταν εκτελεστεί ο πιο κάτω κώδικας;

```
print("************************")
print("*   Υπολογισμός δυνάμεων  *")
print("************************")
print()
x="y"
while x =="y":
  base=int(input("Παρακαλώ δώστε έναν αριθμό"))
  power=int(input("Παρακαλώ δώστε τη δύναμη"))
  print(base**power)
  x=input("Πατήστε y και ENTER")
print("Σας ευχαριστούμε!")
```

3. Να γράψετε ένα πρόγραμμα το οποίο να ζητά απόσταση σε μέτρα και στη συνέχεια να σας εμφανίζει τα εκατοστά (π.χ. 1 μέτρο = 100 εκατοστόμετρα).

4. Να γράψετε ένα πρόγραμμα στο οποίο να προσθέτει 3 αριθμούς. Θα πρέπει να σας ζητά 3 διαφορετικούς αριθμούς και να επαναλαμβάνει τις πράξεις, μέχρι να πατήσετε κάποιο κουμπί εξόδου (π.χ. το "y").

5. Να ξαναγράψετε το πρόγραμμα του παραδείγματος της ενότητας "Φωλιασμένη" επανάληψη. Θα πρέπει να εντοπίζει ο κώδικας κατά πόσο πατήσαμε το "y" ή το "Y" και να τα αναγνωρίζει ως την ίδια επιλογή.

6. Να διορθώσετε τον πιο κάτω κώδικα:

```
print("*   Υπολογισμός διαφοράς  *")
x=int(input("Δώστε έναν αριθμό"
y=int(input(Δώστε ακόμη έναν αριθμό)
while answer=="y
    print(x-y)
    answer=input("Πατήστε y για συνέχεια)
```

5. Συναρτήσεις

```
10 |  print ("Have I gone mad?")
20 |  print ("Am afraid so.  You're totally bonkers!
30 | But let me tell you something.
40 | All great people are!")
50 | #Alice in Wonderland
```

Συναρτήσεις

Η λέξη αυτή είναι σίγουρα ασυνήθιστη… “συναρτήσεις”… κάτι που θα γνωρίσετε στα Μαθηματικά σε μεγαλύτερες τάξεις. Όμως, οι συναρτήσεις (functions, όπως ονομάζονται στα αγγλικά) είναι πάρα πολύ χρήσιμες, όπως θα δούμε στις επόμενες σελίδες.

Συναρτήσεις έχουμε ήδη χρησιμοποιήσει στα παραδείγματα των προηγούμενων κεφαλαίων!

Η print() είναι μια συνάρτηση, που εμφανίζει το περιεχόμενο της παρένθεσης στην οθόνη μας (ή και σε άλλες συσκευές).

Η input() είναι μια άλλη συνάρτηση, που ζητά να δώσουμε δεδομένα (αριθμό, κείμενο) από το πληκτρολόγιο.

Στις επόμενες σελίδες θα μάθουμε περισσότερα για τις συναρτήσεις, και θα δημιουργήσουμε και τις δικές μας!



Τι θα γνωρίσουμε:

Στο **Κεφάλαιο 5 "Συναρτήσεις"** θα γνωρίσουμε τη δομή μιας συνάρτησης. Επίσης θα:

- Δημιουργήσουμε δικές μας συναρτήσεις
- Ενσωματώσουμε συνθήκες σε μια συνάρτηση
- Χρησιμοποιήσουμε επαναλήψεις (loops) σε μια συνάρτηση
- Χρησιμοποιήσουμε συναρτήσεις για δημιουργία σχημάτων στην οθόνη μας



Το παιχνίδι των πράξεων

Οι πύθωνες, όπως γνωρίζουμε, είναι πολύ καλοί στα Μαθηματικά! Μπορούν να θυμηθούν με κάθε λεπτομέρεια ακόμη και πόσα χόρτα και κλαδιά συνάντησαν στον δρόμο τους. Για μας δεν είναι και τόσο σίγουρο, έτσι θα γράψουμε ένα πρόγραμμα για να μας βοηθήσει στις πράξεις των ακέραιων.

```
*testmaths.py - /Users/akoftero/Documents/Development Projects/testmaths.py (3.11.4)*
#Πρόγραμμα για να μάθουμε τη λειτουργία των συναρτήσεων (functions)
#Θα δίνουμε δύο αριθμούς από το πληκτρολόγιο, και η Python θα υπολογίζει
#το άθροισμα ή το γινόμενό τους

print("**************************************************")
print("    Καλωσορίσατε στον αυτόματο υπολογιστή πράξεων!   *")
print("**************************************************")

number1=int(input("Πληκτρολογήστε τον αριθμό 1:"))
number2=int(input("Πληκτρολογήστε τον αριθμό 2:"))
Ln: 9  Col: 6
```

Στον πιο πάνω κώδικα, με σχόλια εξηγούμε τη βασική του λειτουργία. Στη συνέχεια εμφανίζουμε ενημερωτικά μηνύματα στην οθόνη με το print() και ακολούθως, δίνουμε με το πληκτρολόγιο 2 αριθμούς: ο ένας θα είναι η τιμή της μεταβλητής number1, και ο άλλος θα είναι η τιμή της μεταβλητής number2.

Στη συνέχεια, θα εκτελέσουμε τις πράξεις με τους δύο αυτούς αριθμούς.

```
print("Το άθροισμα των αριθμών είναι",number1+number2)
print("Το γινόμενο των αριθμών είναι”,number1*number2)
```

Όταν εκτελέσουμε το πιο πάνω πρόγραμμα, θα πάρουμε το ακόλουθο αποτέλεσμα:

```
IDLE Shell 3.11.4
**************************************************
    Καλωσορίσατε στον αυτόματο υπολογιστή πράξεων!   *
**************************************************
Πληκτρολογήστε τον αριθμό 1:4
Πληκτρολογήστε τον αριθμό 2:3
Το άθροισμα των αριθμών είναι 7
Το γινόμενο των αριθμών είναι 12
>>>
Ln: 15  Col: 0
```

Μέχρι αυτό το σημείο, ο κώδικας μας δε διαφέρει από ό,τι είδαμε μέχρι τώρα. Στη συνέχεια, θα τροποποιήσουμε τις εντολές ώστε να δημιουργήσουμε την πρώτη μας συνάρτηση!



Η πρώτη μας συνάρτηση!

Ορισμένες εντολές μπορούν να ομαδοποιηθούν, ώστε να κάνουν μια συγκεκριμένη εργασία. Έτσι, έχουμε κάποιες εντολές που αφορούν μόνο τον υπολογισμό γινόμενου, ενώ άλλες το άθροισμα δύο αριθμών.

Θα δημιουργήσουμε μια ομάδα εντολών που θα την ονομάσουμε “mymultiplications” και μια ομάδα εντολών που θα την ονομάσουμε “myadditions”. Τα ονόματα δε σημαίνουν κάτι απαραίτητα, φτάνει να ακολουθούμε τους σωστούς κανόνες (όπως είδαμε σε προηγούμενες σελίδες).

Οι ομάδες εντολών που δημιουργούμε, ονομάζονται συναρτήσεις (functions).

Για να δημιουργήσουμε την πρώτη μας συνάρτηση, πρώτα ξεκινάμε με την εντολή def και στη συνέχεια με το όνομα της (mymultiplications) και παρενθέσεις:

```
def mymultiplications():
    print(“Το γινόμενο είναι”,number1*number2)
```

Όλες οι εντολές που βρίσκονται κάτω από τη συνάρτηση (με απόσταση από την αρχή της γραμμής) αποτελούν μέρος της συνάρτησης.

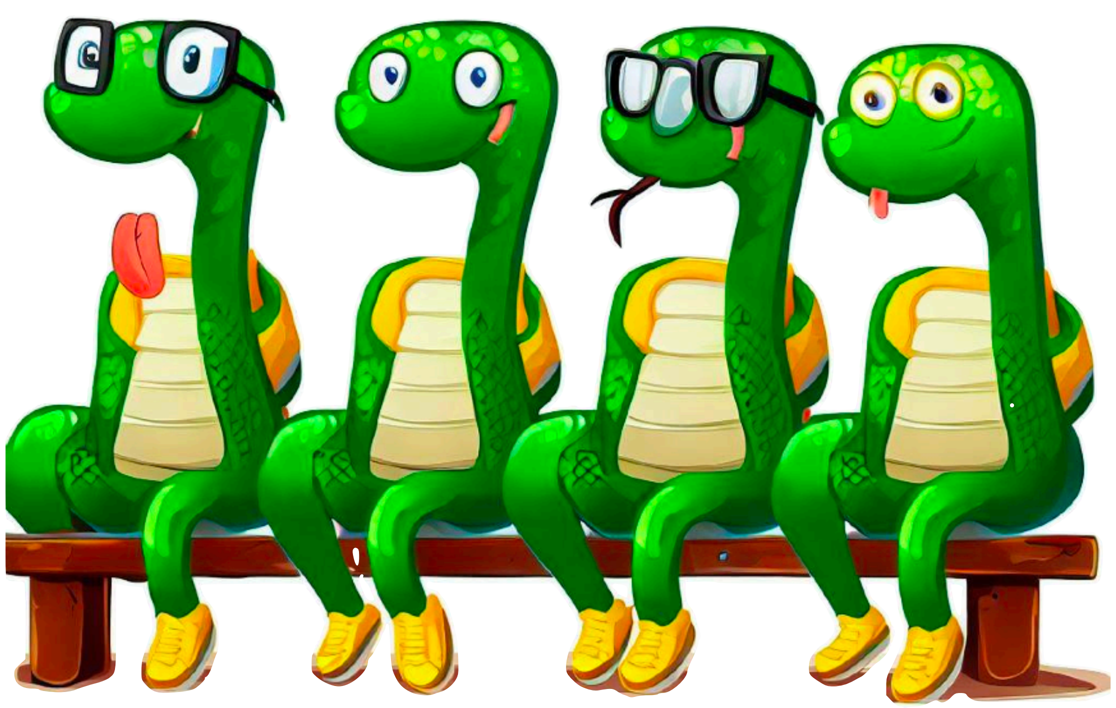
Αν δοκιμάσουμε να εκτελέσουμε το πρόγραμμα, θα δούμε πως δεν εμφανίζεται τίποτα. Αυτό δε σημαίνει πως υπάρχει πρόβλημα με τον κώδικα μας. Αντίθετα! Θα δούμε στη συνέχεια πώς “καλούμε” τις συναρτήσεις!

Οι συναρτήσεις (functions) είναι σαν τους αναπληρωματικούς παίκτες - ξέρουν πολύ καλά τις οδηγίες τους, αλλά περιμένουν υπομονετικά να τους καλέσουμε! Αν δεν τους καλέσουμε, είναι κάπου εκεί, αλλά δεν κάνουν τίποτα.

Για να καλέσουμε μια συνάρτηση, πληκτρολογούμε το όνομά της:

```
mymultiplications()
```

Οι εντολές που περιέχει η συνάρτηση θα εκτελεστούν και το αποτέλεσμα θα φανεί στην οθόνη.



Επιλογή συναρτήσεων

Στη συνέχεια, θα χρησιμοποιήσουμε συνθήκες με την if για να επιλέξουμε ανάμεσα σε συναρτήσεις που θα εκτελέσουμε. Θα δώσουμε δύο αριθμούς από το πληκτρολόγιο. Ακολούθως, θα επιλέξουμε αν θα κάνουμε πρόσθεση ή πολλαπλασιασμό.

```
num1=int(input("Παρακαλώ δώστε έναν αριθμό"))
num2=int(input("Παρακαλώ δώστε δεύτερο αριθμό"))
```

Με τις δύο πιο πάνω εντολές, δίνουμε δύο αριθμούς από το πληκτρολόγιο.

```
def mymultiplication():
    print("Το γινόμενο είναι",num1*num2)
```

Με την πιο πάνω συνάρτηση, υπολογίζουμε το γινόμενο των δύο αριθμών.

```
def sum():
    print("Το άθροισμα είναι",num1+num2)
```

Με τη συνάρτηση sum() υπολογίζουμε το άθροισμα των δύο αριθμών.

Στη συνέχεια, θα δώσουμε επιλογές ώστε να επιλέξουμε αν θέλουμε να κάνουμε πρόσθεση ή αφαίρεση:

```
choice=input("Παρακαλώ πατήστε: a για πρόσθεση, b για πολλαπλασιασμό")
```

Με την πιο πάνω εντολή, πληκτρολογούμε είτε το a είτε το b, ώστε να επιλέξουμε το είδος της πράξης που θα κάνουμε.

```
if choice=="a":
    sum()
if choice=="b":
    mymultiplication()
```

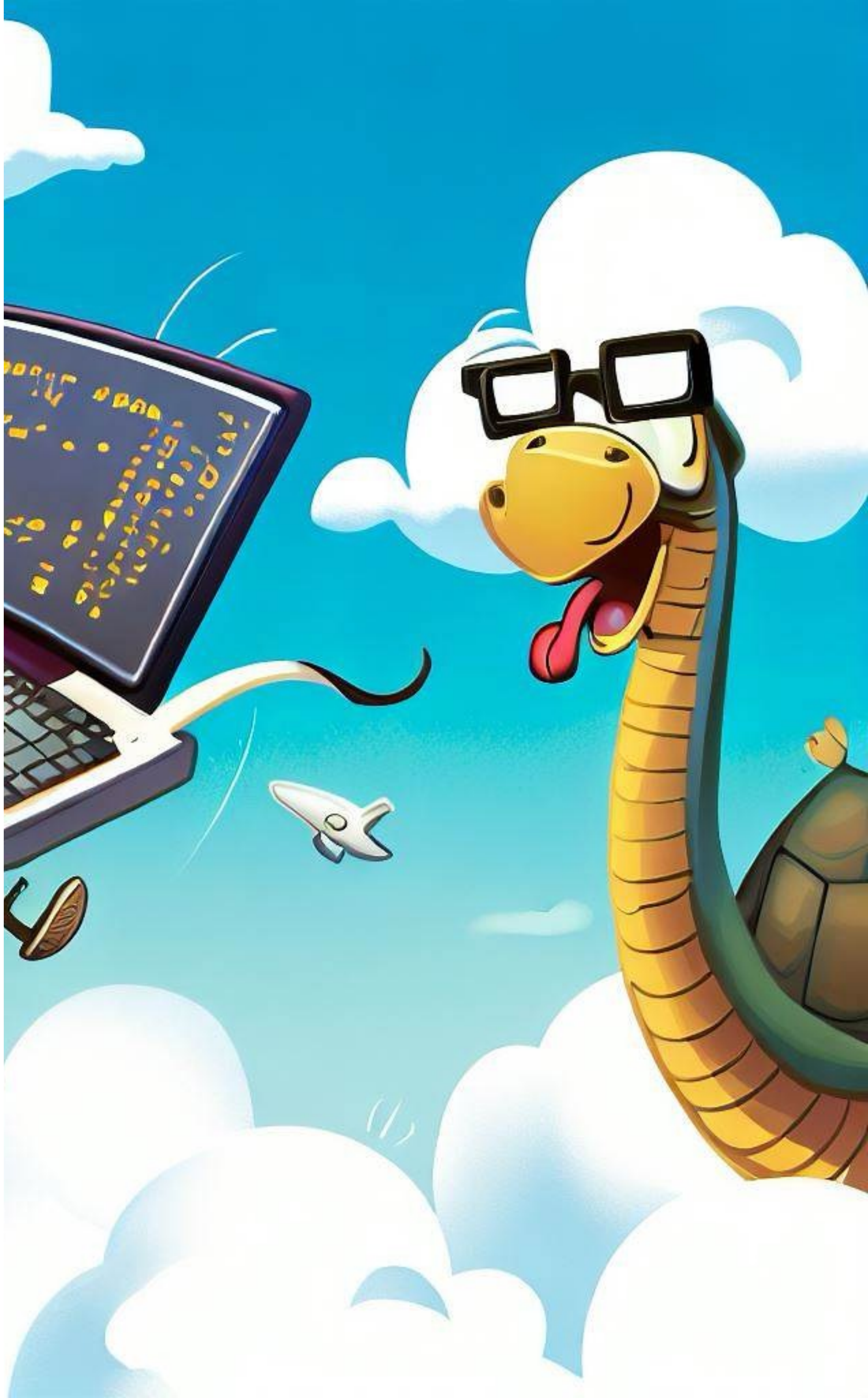
Με τις πιο πάνω εντολές, μπορούμε να επιλέξουμε αν θα καλέσουμε τη συνθήκη με την οποία θα υπολογιστεί το άθροισμα ή τη συνθήκη με την οποία θα υπολογιστεί το γινόμενο των δύο αριθμών.

Με τις συναρτήσεις θα δημιουργήσουμε πολύπλοκα γεωμετρικά σχήματα, με τη βοήθεια του module "turtle" για Python. Περισσότερα για τις συναρτήσεις και τα γεωμετρικά σχήματα θα γνωρίσουμε στο **Μέρος Γ', "Χελώνες και Πύθωνες".**



Τι μάθαμε μέχρι τώρα

- Στο Κεφάλαιο 5 γνωρίσαμε τις συναρτήσεις
- Δημιουργήσαμε δική μας συνάρτηση με την εντολή def
- Καλέσαμε μέσα στον κώδικα μας τη συνάρτηση που δημιουργήσαμε
- Συνδυάσαμε επαναλήψεις με τις συναρτήσεις μας



Δραστηριότητες

1. Δοκιμάστε, μέσα σε μια συνάρτηση, να καλέσετε την ίδια. Τι νομίζετε ότι θα συμβεί; Δοκιμάστε να το εξηγήσετε!

```
def mymultiplications():
    print(“Hello”)
    mymultiplications()

mymultiplications()
```

2. Να βρείτε τα λάθη της πιο κάτω συνάντησης και να τη διορθώσετε. Τι θα εμφανιστεί στην οθόνη όταν εκτελεστεί;

```
def division()
    print(“Το πηλίκο είναι, 25/5)
    divisions()

divisions()
```

3. Χωρίς να τρέξετε τον πιο κάτω κώδικα, να γράψετε το αποτέλεσμα που θα έχει στην οθόνη σας:

```
print("****************************")
print("*    Υπολογισμός Γινομένου.    *")
print("****************************")
print()

#---Συνάρτηση-----

def mymultiplications():
    print(“Το γινόμενο είναι”, number1*number2)

#---Συνάρτηση-----

number1=int(input(“Δώστε έναν αριθμό”))
number2=int(input(“Δώστε ακόμη έναν αριθμό”))

mymultiplications() #καλούμε τη συνάρτηση
```

4. Να γράψετε ένα πρόγραμμα το οποίο να σας ζητά έναν αριθμό και στη συνέχεια να επιλέγετε αν (α) θα προβάλει τον πίνακα πολλαπλασιασμού του (μέχρι το 12) ή αν θα δείχνει τον αριθμό στη δεύτερη δύναμη (το γινόμενο με τον εαυτό του).



ΜΕΡΟΣ Γ΄: ΧΕΛΩΝΕΣ & ΠΥΘΩΝΕΣ!

ΓΡΑΦΙΚΑ ΧΕΛΩΝΑΣ!

Όσο παράξενο και αν ακούγεται, οι Πύθωνες μπορούν να σχεδιάσουν! Οκ, ίσως όχι οι πραγματικοί -και δε θέλουμε να πλησιάσετε πολύ για να το διαπιστώσετε μόνοι σας!- αλλά οι ψηφιακοί σίγουρα! Για την ακρίβεια, δανείζονται τα "εργαλεία" από τις φίλες τους τις χελώνες.

Στο Μέρος Γ' του βιβλίου θα ασχοληθούμε με τις εντολές που επιτρέπουν στην Python να δημιουργήσει σχήματα. Συγκεκριμένα, θα χρησιμοποιήσουμε το άρθρωμα (παράξενη λέξη, σωστά;) με τις εντολές που επιτρέπουν τη δημιουργία σχημάτων με γραμμές.

Ακολούθως, θα χρησιμοποιήσουμε τις εντολές αυτές για να δημιουργήσουμε γεωμετρικά σχήματα!



Τι θα γνωρίσουμε:

Μέσα από το **Μέρος Γ' "Χελώνες και Πύθωνες"** θα έχουμε την ευκαιρία:

- Να γνωρίσουμε και να εισαγάγουμε το άρθρωμα turtle.
- Να γνωρίσουμε τις βασικές εντολές δημιουργίας σχημάτων.
- Να δημιουργήσουμε τα δικά μας σχήματα με χρήση επαναλήψεων.
- Να κατανοήσουμε τον τρόπο με τον οποίο "κινείται" η χελώνα μας στην οθόνη.
- Να αξιοποιήσουμε τις συναρτήσεις για να δημιουργήσουμε πολύπλοκα γεωμετρικά σχήματα.



Αρθρώματα της Python

Οι δυνατότητες της Python είναι σχεδόν απεριόριστες! Αυτό οφείλεται (και) στο ότι μπορούμε να χρησιμοποιήσουμε αρθρώματα (modules) για να δώσουμε πολλές και νέες δυνατότητες στα προγράμματα που δημιουργούμε, χωρίς να χρειαστεί να γράψουμε πολύπλοκο κώδικα. Οκ, σίγουρα θα γράψουμε αρκετό κώδικα, αλλά ας τον κρατήσουμε όσο πιο απλό γίνεται!

Το άρθρωμα turtle

Ένα χρήσιμο άρθρωμα (module) είναι το turtle (χελώνα). Βασίζεται στο περιβάλλον προγραμματισμού LOGO που δημιουργήθηκε το 1967 για να διδάξει μαθηματικά σε παιδιά στις ΗΠΑ (με τη χρήση μιας χελώνας ρομπότ!).

Αν έχετε εγκαταστήσει τη Python με τις οδηγίες αυτού του βιβλίου, τότε έχετε στον υπολογιστή σας και το άρθρωμα turtle.

Η λογική της LOGO -και στα παραδείγματα μας, της Python με τις πρόσθετες εντολές- είναι απλή: δίνουμε εντολές (FORWARD, LEFT, RIGHT, κτλ) σε μια "χελώνα" στην οθόνη μας. Η "χελώνα" κινείται πάνω στην οθόνη και δημιουργεί σχέδια. Τα (γεωμετρικά) αυτά σχέδια μπορούν να έχουν διαφορετικό σχήμα (περιγράμματος), χρώμα γραμμής κ.α. Οι δυνατότητες που προσφέρει το άρθρωμα turtle είναι (σχεδόν) απεριόριστες, και θα γνωρίσουμε αρκετές από αυτές.

Το πρώτο που πρέπει να κάνουμε, είναι να εισαγάγουμε το άρθρωμα turtle:

```
from turtle import *
```

Στη συνέχεια πληκτρολογούμε την πρώτη μας εντολή για δημιουργία σχήματος:

```
forward(100)
```

Με το πιο πάνω θα δημιουργηθεί μια γραμμή με μήκος 100 εικονοστοιχεία (pixels).

Θα γνωρίσουμε περισσότερα για τα εικονοστοιχεία (pixels) και τις εντολές δημιουργίας σχημάτων στη συνέχεια!



Εικονοστοιχεία (Pixels!)

Η οθόνη κάθε ψηφιακής συσκευής, από τον υπολογιστή μέχρι το κινητό μας τηλέφωνο και το smart watch, αποτελείται από πολύ μικρά τετραγωνάκια - τα εικονοστοιχεία (Pixels, από τις λέξεις Picture Elements).



Οι σύγχρονες οθόνες έχουν τόσες πολλές γραμμές και στήλες με pixels (η “ανάλυση” που λέμε…) που δεν μπορούμε εύκολα να τα ξεχωρίσουμε. Βέβαια, σε παλαιότερες εποχές όπου οι οθόνες είχαν πολύ λιγότερα pixel, τα τετραγωνάκια της οθόνης φαίνονταν πολύ καθαρά. Και τα γραφικά ήταν πολύ πιο ‘τετραγωνισμένα’.

Όταν δίνουμε την εντολή forward(100), ζητάμε από τη “χελώνα” μας να προχωρήσει προς την κατεύθυνση στην οποία “κοιτάζει” κατά 100 pixels.

Κάθε οθόνη έχει διαφορετική ανάλυση. Και δεν εξαρτάται πάντα από το μέγεθος της. Για παράδειγμα, ένας παλαιότερος υπολογιστής με 17’’ οθόνη, ίσως έχει πολύ μικρότερη ανάλυση από έναν σύγχρονο φορητό υπολογιστή με 14’’ οθόνη.



Αν το tablet ή το laptop σας έχει οθόνη HD, αυτό σημαίνει πως αποτελείται από 1920 στήλες και 1080 γραμμές με pixels. Αν θέλουμε να μάθουμε πόσα pixels είναι αυτό συνολικά, μπορούμε να χρησιμοποιήσουμε την υπολογιστική μας μηχανή, ή μολύβι και χαρτί ή την ίδια την Python!

```
>>> 1920*1080
    2073600
```

Χρώματα & Γραμμές!

Τι γίνεται με τα χρώματα; Καλά τα σχήματα, αλλά θέλουμε και λίγο χρώμα στις τάξεις μας, στη γειτονιά μας, στα βιβλία μας, στις οθόνες μας!

Πριν από τις εντολές σχεδίασης, μπορούμε να επιλέξουμε ένα χρώμα (στη συνέχεια θα δούμε και πώς αλλάζουμε χρώματα στην πορεία).

```
from turtle import *
color(“blue”) #το χρώμα γίνεται μπλε
forward(100)
```

Στην εντολή color() βάζουμε στην παρένθεση το χρώμα που θέλουμε να χρησιμοποιήσουμε στη δημιουργία ενός σχήματος. Για το πιο πάνω παράδειγμα, δημιουργούμε μια μπλε γραμμή μήκους 100 pixels.

Ας δοκιμάσουμε διάφορους συνδυασμούς:

```
from turtle import *
color(“blue”) #το χρώμα γίνεται μπλε
forward(100)
color(“green”)
forward(100)
```

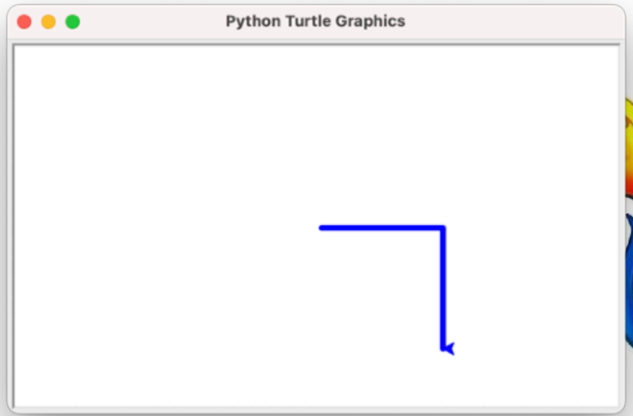
Διαπιστώνουμε εύκολα πως η αλλαγή χρώματος είναι μια σχετικά απλή διαδικασία.



Πριν την εντολή με την οποία θα δημιουργηθεί το σχήμα, επιλέγουμε και το χρώμα. Το χρώμα μπορούμε να το επιλέξουμε με την ονομασία του: black (μαύρο), blue (κυανό), red (κόκκινο), yellow (κίτρινο), pink (ροζ), brown (καφέ), gold (χρυσό), maroon (κεραμιδί), κ.α.

Για να αλλάξουμε το πλάτος της γραμμής, χρησιμοποιούμε την εντολή width() με έναν αριθμό:

```
from turtle import *
color(“blue”)   #το χρώμα γίνεται μπλε
width(5)        #το πλάτος της γραμμής γίνεται 5
forward(100)
right(90)
forward(100)
```



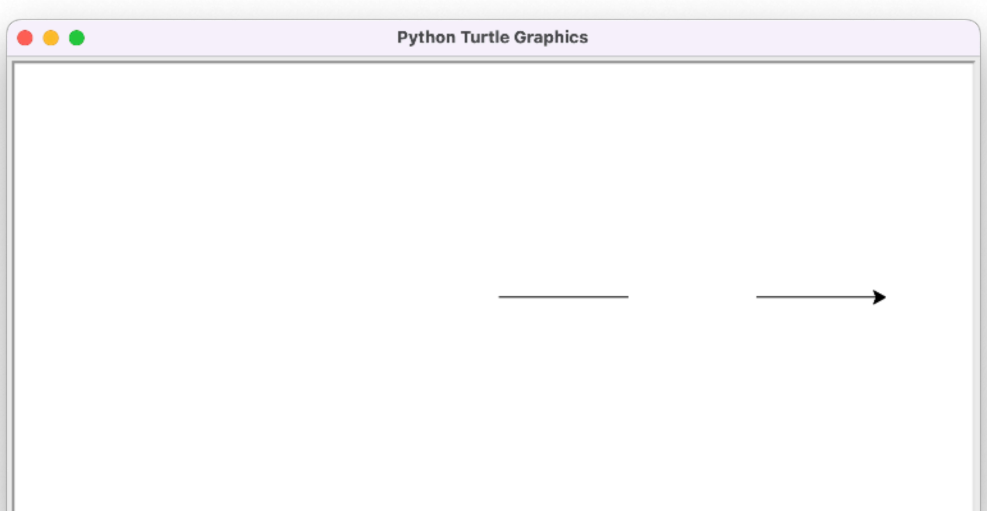


Πένες πάνω και κάτω...

Αν θέλουμε να δημιουργήσουμε μια γραμμή μήκους 200px, θα δώσουμε την εντολή forward(200). Αν, όμως, θέλουμε να δημιουργήσουμε μια σειρά από γραμμές με αποστάσεις μεταξύ τους, τότε θα πρέπει να μάθουμε στη χελώνα να "σηκώνει" την πένα καθώς προχωρά.

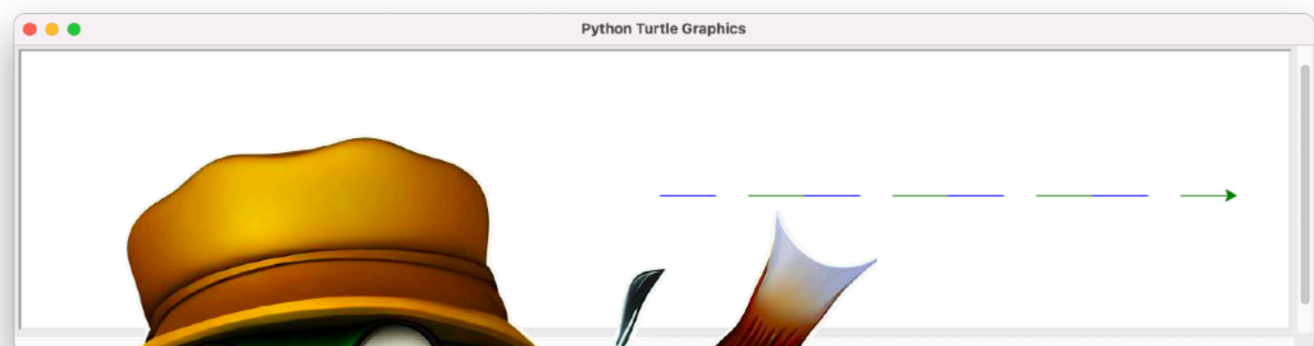
Με το penup(), λέμε στη χελώνα μας να "σηκώσει" την πένα. Αμέσως μετά ακολουθεί η εντολή κίνησης, ώστε να μετακινηθεί σε άλλη θέση. Αν θέλουμε να αρχίσει να γράφει και πάλι, τότε δίνουμε την εντολή pendown().

```
from turtle import *
forward(100) #προχωρά 100px
penup() #η πένα “σηκώνεται” και δεν γράφει!
forward(100) #προχωρά 100px
pendown() #η πένα “κάθεται” στο χαρτί.
forward(100)
```



Ας βάλουμε λίγο χρώμα (και επανάληψη) στον κώδικα:

```
from turtle import *
for counter in range(0, 4):
    color(“blue”)
    forward(50)
    penup()
    color(“green”)
    forward(30)
    pendown()
    forward(50)
```



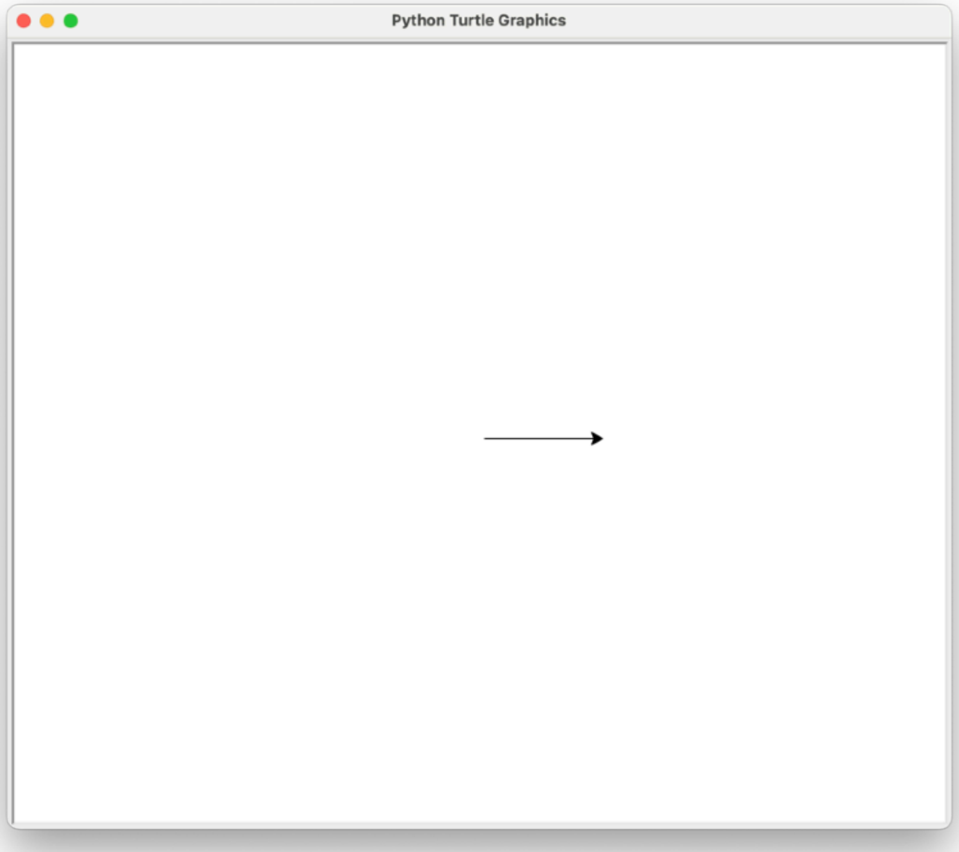


Πύθωνες & Γεωμετρία

Με τη βοήθεια της Python μπορούμε να δημιουργήσουμε όλα τα γεωμετρικά σχήματα! Ας δούμε παραδείγματα:

```
from turtle import *
forward(100)
```

Εκτελούμε το πρόγραμμα με Run και εμφανίζεται η πρώτη μας γραμμή. Επειδή η χελώνα μας (έτσι θα αποκαλούμε πλέον το βελάκι) "κοιτάζει" δεξιά, η γραμμή μας σχηματίζεται προς αυτή την κατεύθυνση.



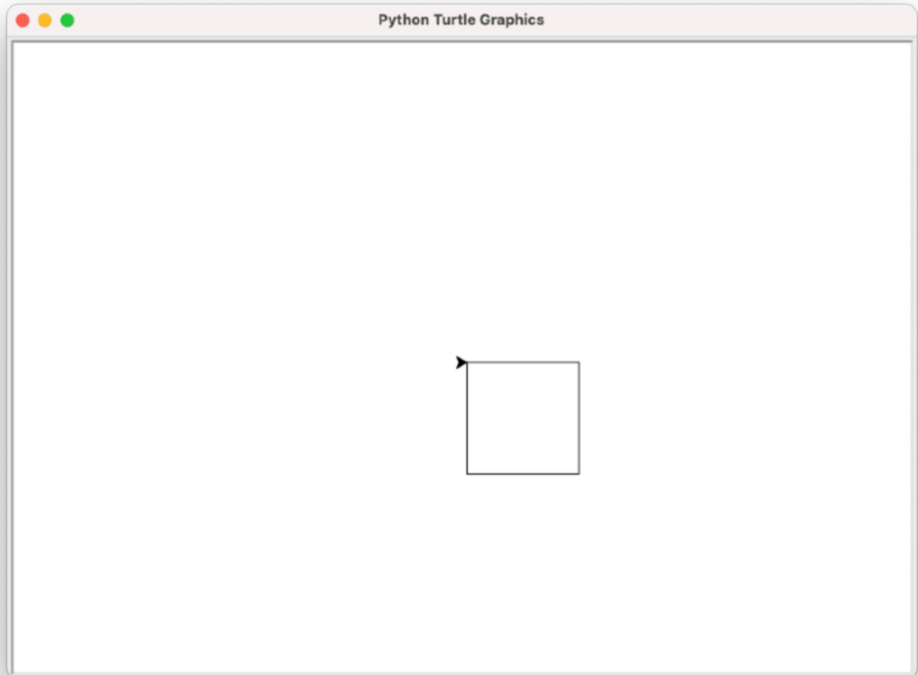
Ας δημιουργήσουμε ένα τετράγωνο: θα ξεκινήσουμε με δύο βασικές εντολές: με τη μία θα προχωράμε μπροστά (δεξιά, όπως δείχνει η χελώνα) και με την άλλη θα κάνουμε στροφή 90 μοιρών προς τα δεξιά.

```
from turtle import *
forward(100)
right(90)
```

Στη forward(100), ο αριθμός στην παρένθεση δηλώνει τα "βήματα" που θα κάνει (σε pixels). Στην εντολή right(90), ο αριθμός στην παρένθεση είναι οι μοίρες που θα στρίψει (προς τα δεξιά).

```
from turtle import *
forward(100)
right(90)
forward(100)
right(90)
forward(100)
right(90)
forward(100)
right(90)
```

Στην επόμενη σελίδα βλέπουμε το αποτέλεσμα της εκτέλεσης του πιο πάνω κώδικα.



pythongeometry.py - /Users/akoftero/Documents/pythongeometry.py (3.11.4)

```
from turtle import *
x=0
while x<4: #ελέγχει αν το χ είναι μικρότερο του 4
    forward(100)
    right(90)
    x+=1 #θα μπορούσαμε να γράφαμε και χ=χ+1
```

Ln: 3 Col: 49

Η χελώνα μας δημιουργεί σιγά σιγά το σχήμα, όπως το βλέπουμε και στην εικόνα πιο πάνω.

Ο κώδικας που είδαμε στην προηγούμενη σελίδα, δημιουργεί το τετράγωνο της εικόνας πάνω. Είναι σημαντικό να μπορούμε να εντοπίζουμε τμήματα του κώδικα που θα μπορούσαμε να αλλάξουμε ή/και να βελτιώσουμε. Για παράδειγμα, παρατηρούμε ότι επαναλαμβάνουμε 4 φορές τις εντολές forward(100) και right(90).

Θα χρησιμοποιήσουμε το while() για να απλοποιήσουμε τον κώδικα μας: θα πρέπει η επανάληψη να γίνει 4 φορές.

Γι'αυτό είναι απαραίτητο να χρησιμοποιήσουμε μια μεταβλητή x που θα μετράει τις επαναλήψεις!

Αν εκτελέσουμε το πιο πάνω πρόγραμμα, θα δούμε ότι δημιουργεί το τετράγωνο της εικόνας αριστερά. Απλά έχουν μειωθεί οι εντολές.

Στον κώδικα, είναι σημαντικό να εντοπίζουμε τα στοιχεία που μπορούμε να απλοποιήσουμε. Δεν είναι πάντα απαραίτητο (ή και χρήσιμο) να έχουμε επαναλήψεις. Όμως, είναι πολύ σημαντικό, όταν γράφουμε κώδικα, να εντοπίζουμε "μοτίβα" που θα μπορούσαμε να απλοποιήσουμε.

Το τετράγωνο είναι μια ιδιαίτερη περίπτωση. Σίγουρα θα ήταν δυσκολότερο με ένα ορθογώνιο σχήμα.

Επανάληψη με For...

Εκτός από το while(), μπορούμε να χρησιμοποιήσουμε και την εντολή for για επανάληψη, όπως είδαμε σε προηγούμενο κεφάλαιο.

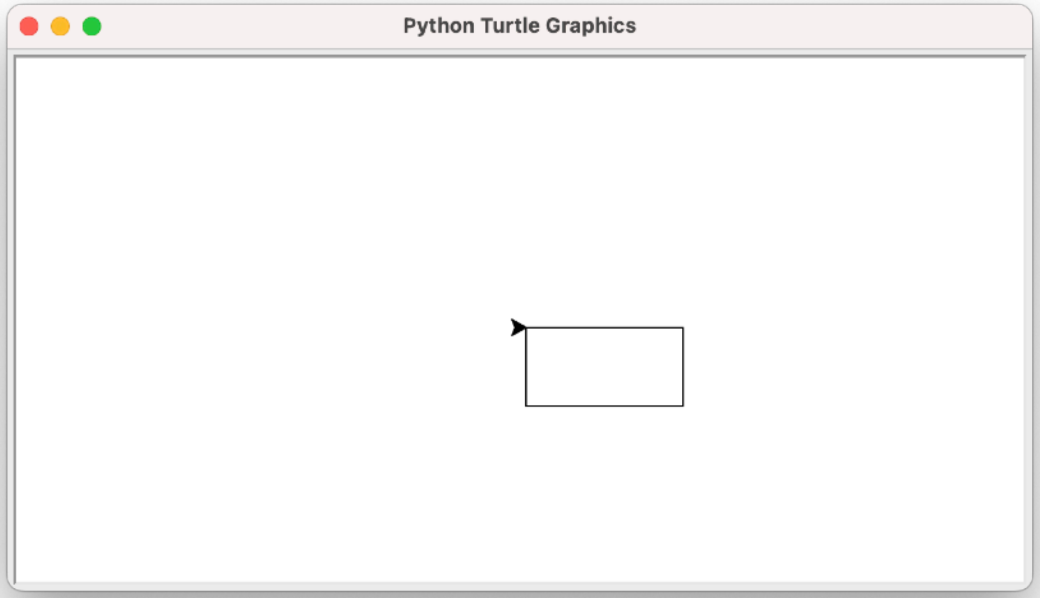
```
from turtle import *
for x in range(0,4):
    forward(100)
    right(90)
```

Η εντολή for ξεκινά με μια μεταβλητή (εδώ χρησιμοποιήσαμε τη μεταβλητή x) η οποία παίρνει τιμές από το 0 μέχρι το 4 (range).

Στη συνέχεια, εκτελεί τις εντολές που βρίσκονται μέσα στη δομή επανάληψης, όσες φορές είναι και το εύρος (range).

Θα χρησιμοποιήσουμε τη for για να δημιουργήσουμε ένα ορθογώνιο σχήμα με μήκος 100 pixels και πλάτος 50 pixels. Στο ορθογώνιο, επειδή η μια πλευρά είναι διαφορετική από την άλλη, δεν μπορούμε να επαναλάβουμε 4 φορές το ίδιο ζευγάρι εντολών. Θα επαναλάβουμε 2 φορές, δύο ζευγάρια:

```
for x in range(0,2):
    forward(100)
    right(90)
    forward(50)
    right(90)
```



Σχεδόν σε κάθε σχήμα μπορούμε να εντοπίσουμε μοτίβα για χρήση επαναλήψεων. Πολλές φορές μπορεί να δυσκολευόμαστε να εντοπίσουμε από την αρχή το μοτίβο. Μπορούμε να γράψουμε -έστω σε χαρτί- τις εντολές που θα εκτελεί το πρόγραμμα μας, ώστε να βρούμε το μοτίβο (αν υπάρχει) και να χρησιμοποιήσουμε την επανάληψη.

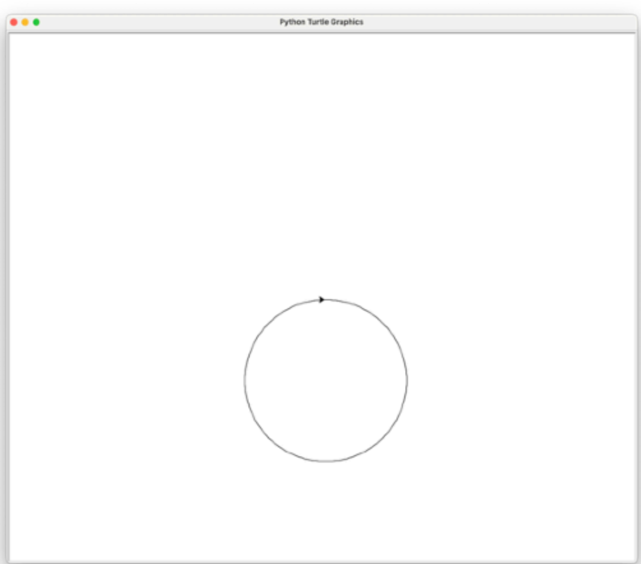


Τρέχοντας σε κύκλους

Θα δημιουργήσουμε έναν κύκλο. Απαραίτητα θα πρέπει να χρησιμοποιήσουμε επανάληψη.

```
from turtle import *
for x in range(0,180):
    forward(5)
    right(2)
```

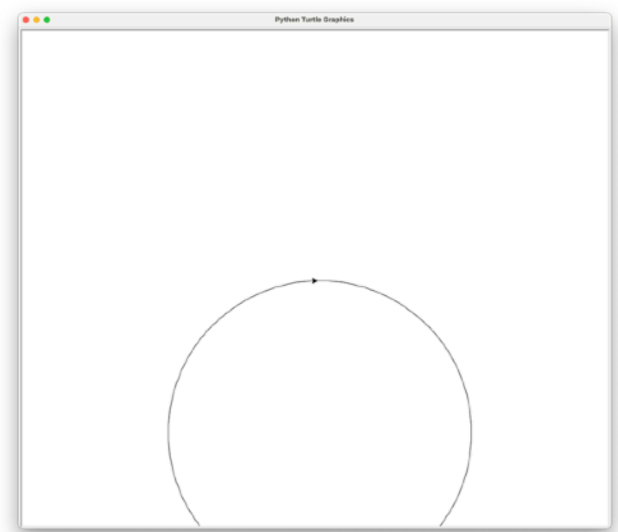
Ο κύκλος έχει 360 μοίρες. Στο range δώσαμε μέγιστη τιμή το 180. Στη στροφή, δώσαμε εντολή να πηγαίνει δεξιά 2 μοίρες. Για να κλείσει ο κύκλος, το γινόμενο της τιμής στο range (180) με την τιμή στο right (ή left) πρέπει να ισούται με 360. (180X2=360).



Τι θα συμβεί αν αλλάξουμε το forward; Από 5 το διπλασιάζουμε σε 10.

```
from turtle import *
for x in range(0,180):
    forward(10) #το αυξήσαμε από 5
    right(2)
```

Ο κύκλος μας έχει διπλασιαστεί (περιφέρεια). Με παρόμοιο τρόπο μπορούμε να παίξουμε με τις υπόλοιπες παραμέτρους (π.χ. το range να γίνει 90 και η στροφή να γίνει 4 μοίρες). Μπορούμε επίσης να κάνουμε αριστερή στροφή (left) αντί για στροφή προς τα δεξιά.

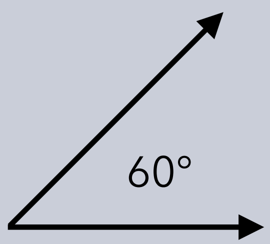


Μπορούμε να δημιουργήσουμε κύκλο και με τη συνάρτηση circle(). Μέσα στην παρένθεση γράφουμε το μήκος της ακτίνας. Οι εντολές είναι:

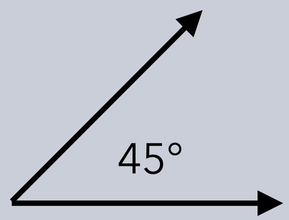
```
from turtle import *
circle(10)
```

Είδη γωνιών

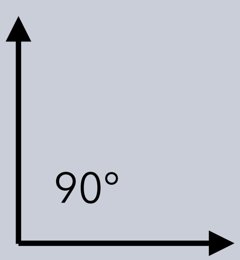
Το άνοιγμα των γωνιών το μετρούμε σε μοίρες. Τις γωνίες τις μετράμε με ένα ειδικό εργαλείο, το μοιρογνωμόνιο . Ανάλογα με το άνοιγμα τους, τις χωρίζουμε σε διάφορες ομάδες. Για παράδειγμα, οι γωνίες με άνοιγμα μεγαλύτερο από 0° και μικρότερο από 90° ονομάζονται οξείες (εικόνες πιο κάτω).



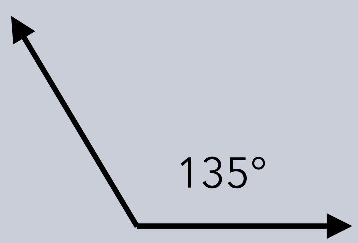
Οξεία γωνία



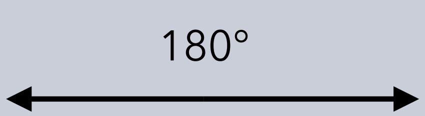
Οξεία γωνία



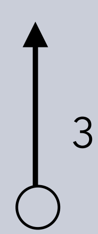
Ορθή γωνία



Αμβλεία γωνία



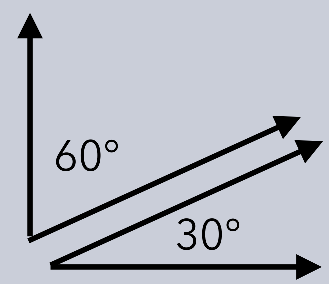
Ευθεία γωνία

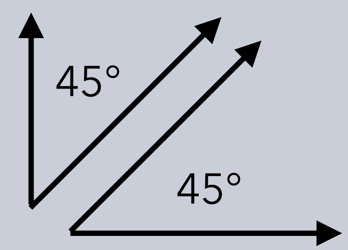


Πλήρης γωνία

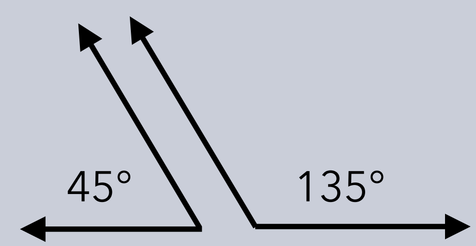
Συμπληρωματικές & Παραπληρωματικές

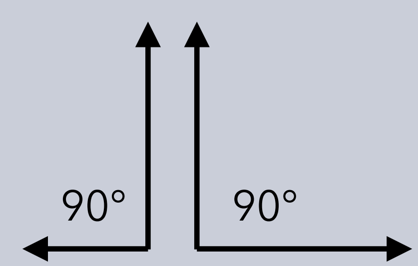
Μια γωνία με άνοιγμα 90° ονομάζεται ορθή. Μια γωνία με άνοιγμα 180° ονομάζεται ευθεία. Αν έχουμε μια γωνία 30° και θέλουμε να ενωθεί με μιαν άλλη για να γίνουν μαζί 90°, η άλλη γωνία πρέπει να έχει άνοιγμα 60°. Οι δύο αυτές γωνίες, που το άθροισμά τους κάνει 90° (μια ορθή γωνία) ονομάζονται συμπληρωματικές γωνίες.





Δύο γωνίες που το άθροισμά τους είναι 180° (ευθεία γωνία), ονομάζονται παραπληρωματικές.





Δημιουργία τριγώνου

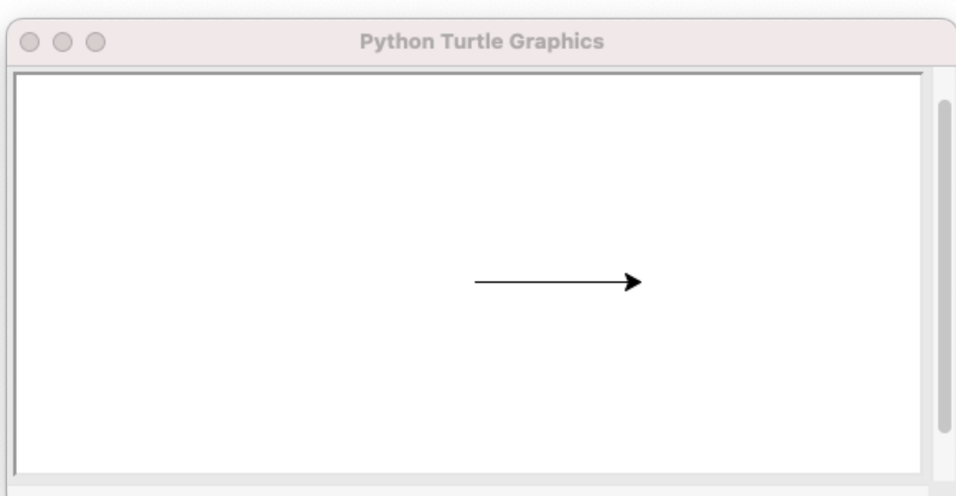
Για να δημιουργήσουμε τρίγωνα με Python, θα πρέπει να έχουμε κατανοήσει τα είδη των γωνιών και κυρίως τις συμπληρωματικές και παραπληρωματικές γωνίες.

Ας δούμε το ακόλουθο παράδειγμα: θέλουμε να δημιουργήσουμε ένα ισόπλευρο τρίγωνο, με μήκος κάθε πλευράς 100px (pixels). Το ισόπλευρο έχει όλες τις γωνίες του ίσες (60°).

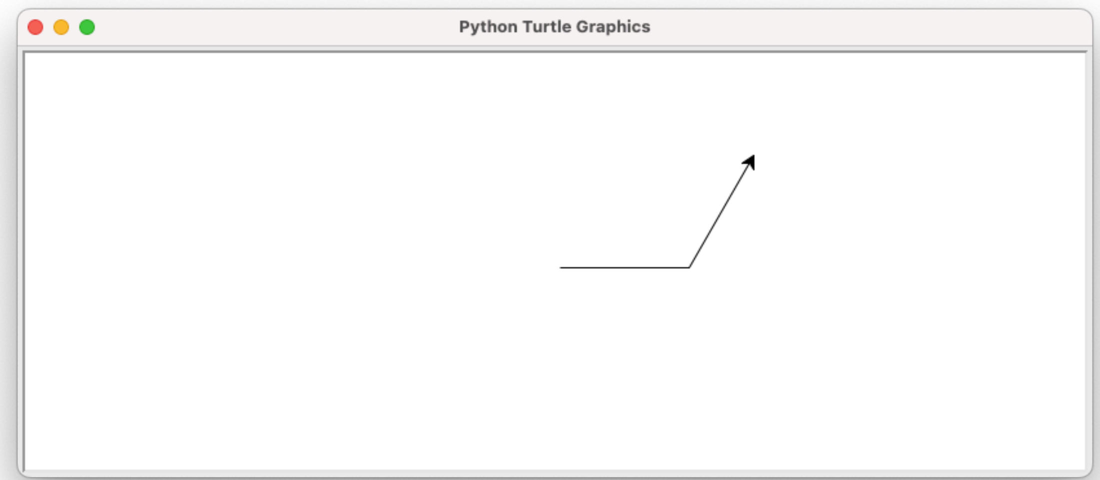
Με την πιο κάτω εντολή, δημιουργούμε την βάση (κάτω πλευρά) του τριγώνου:

```
from turtle import *
forward(100)
```

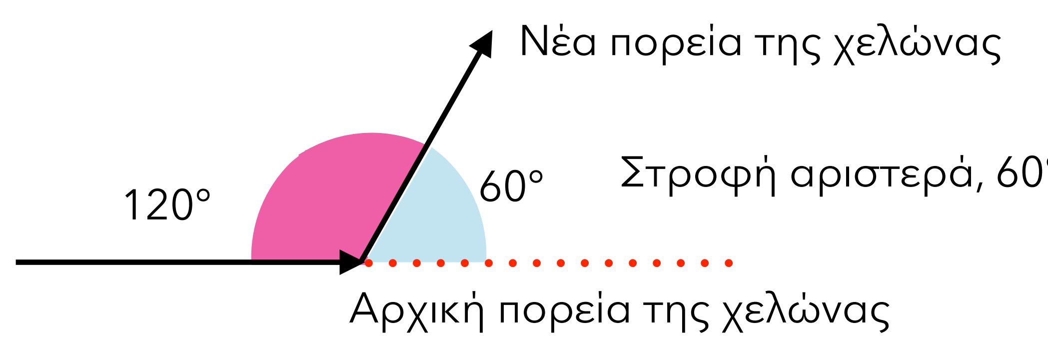
Επειδή η χελώνα μας βλέπει πάντοτε δεξιά, θα δούμε να εμφανίζεται αυτό:



Όπως είπαμε στην αρχή, σε ένα ισόπλευρο τρίγωνο, κάθε γωνία έχει άνοιγμα 60°. Αν δοκιμάσουμε να δώσουμε την εντολή left(60) και στη συνέχεια forward(100), θα δούμε το εξής:



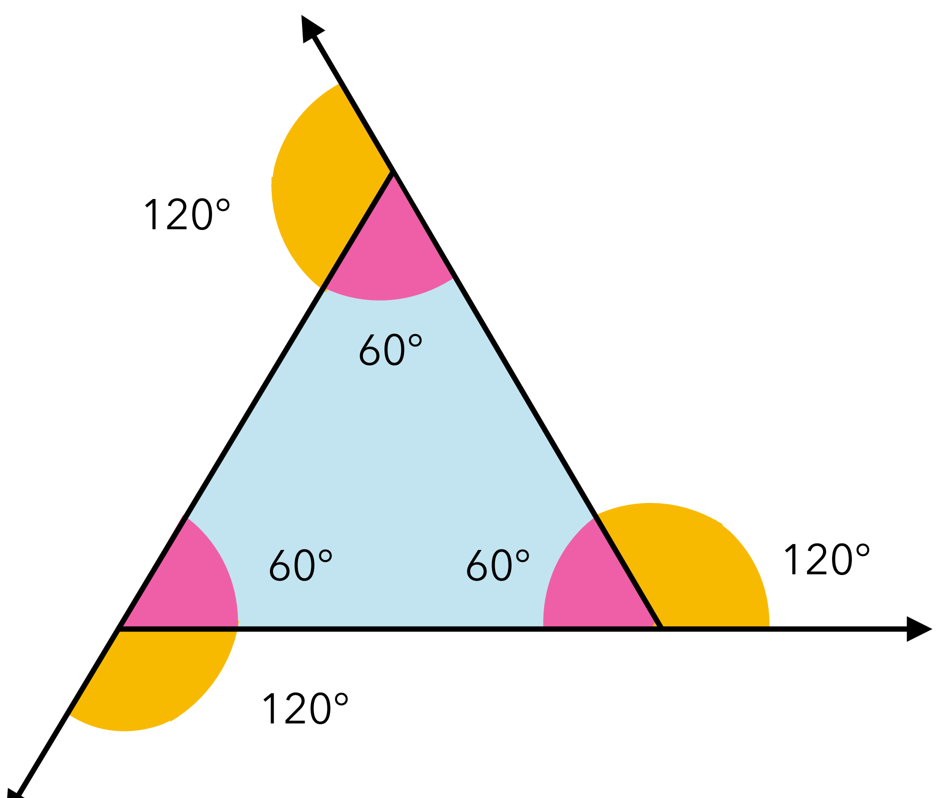
Η γωνία μας είναι εντελώς λανθασμένη (ή, για την ακρίβεια, δεν είναι αυτή που θέλαμε). Και δεν είναι καν 60°, είναι 120°! Ο λόγος είναι απλός: Η χελώνα μας, “κοίταζε” δεξιά. Όταν της είπαμε να στρίψει 60° αριστερά, αυτό έκανε! Στην πραγματικότητα, δημιουργήθηκαν δύο γωνίες (παραπληρωματικές).



Στο παράδειγμα της προηγούμενης σελίδας, βλέπουμε το λάθος στη στροφή - η χελώνα μας δεν είναι αρκετά "έξυπνη" για να καταλάβει ότι θέλουμε να κατασκευάσουμε τρίγωνο.

Το "μυστικό" είναι να σκεφτόμαστε πάντα με τις παραπληρωματικές (και σε κάποιες περιπτώσεις τις συμπληρωματικές) γωνίες: "πόσες μοίρες να στρίψουμε ώστε η γωνία που θα σχηματιστεί εσωτερικά να είναι αυτή που θέλουμε;"

Στην περίπτωση του τριγώνου μας, θέλουμε η γωνία (εσωτερικά) να είναι 60°, άρα εμείς θα πρέπει να στρίψουμε αριστερά 120°, όση είναι δηλαδή η παραπληρωματική της γωνία.

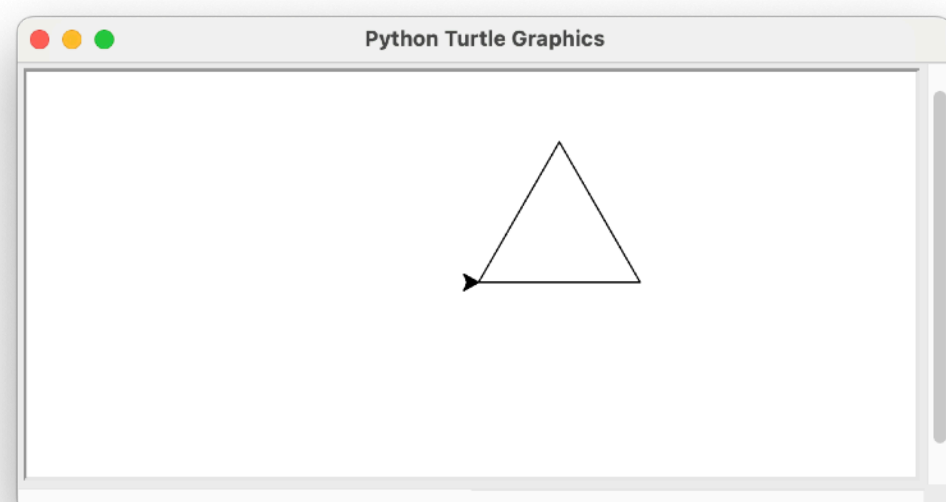


Ο κώδικας για το ισόπλευρο τρίγωνο είναι:

```
from turtle import *
forward(100)
left(120)
forward(100)
left(120)
forward(100)
left(120)
```

Οι εντολές forward() και left() επαναλαμβάνονται 3 φορές. Μπορούμε να χρησιμοποιήσουμε μια επανάληψη:

```
from turtle import *
for x in range(0,3):
    forward(100)
    left(120)
```



Θέση χελώνας στην οθόνη

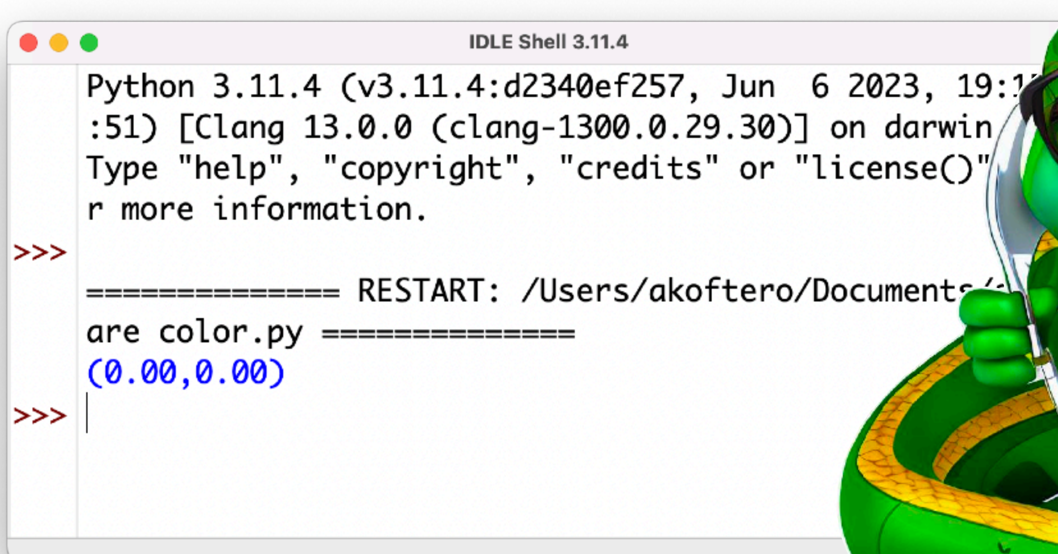
Η οθόνη του υπολογιστή αποτελείται από σειρές και στήλες με pixels, όπως είδαμε σε προηγούμενη σελίδα (“Εικονοστοιχεία”).

Η “χελώνα” μας εμφανίζεται πάντοτε στο κέντρο της οθόνης (του παραθύρου για την ακρίβεια) και “κοιτάζει” δεξιά. Η θέση αυτή ονομάζεται “home”. Έχει, επίσης, τις συντεταγμένες (0,0).

Αν θέλουμε να δούμε, ανά πάσα στιγμή, τη θέση της χελώνας, χρησιμοποιούμε την πιο κάτω εντολή:

```
print(pos())
```

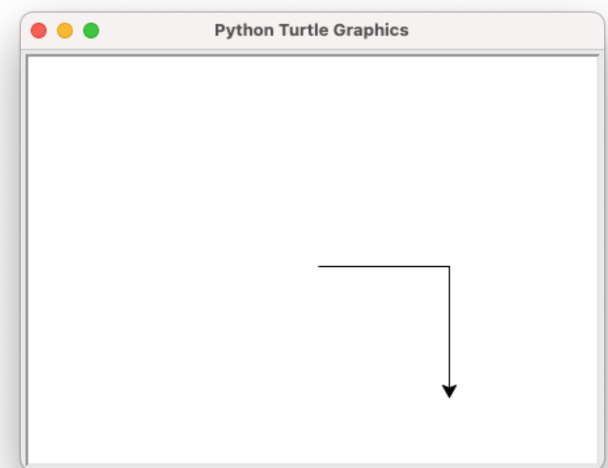
Δε θα εμφανιστεί η πληροφορία στο παράθυρο εμφάνισης των γραφικών, αλλά στο παράθυρο IDLE της Python (εικόνα πιο κάτω).



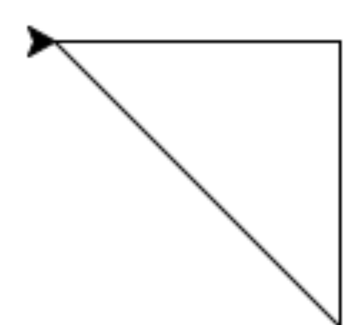


Με την home(), στέλνουμε τη χελώνα μας στην αρχική της θέση! Προσοχή όμως! Δε θα μετακινηθεί απλά, αλλά θα τραβήξει μια γραμμή από εκεί που βρίσκεται μέχρι την αρχική της θέση!

```
forward(100) #σχεδιάζει γραμμή δεξιά, 100px
right(90) #δεξιά στροφή 90°
forward(100) #σχεδιάζει γραμμή προς τα κάτω, 100px
```



Αν δώσουμε και την εντολή home(), τότε η χελώνα θα μετακινηθεί από το σημείο που βρίσκεται στην αρχική της θέση, και θα ενώσει τα δύο σημεία.

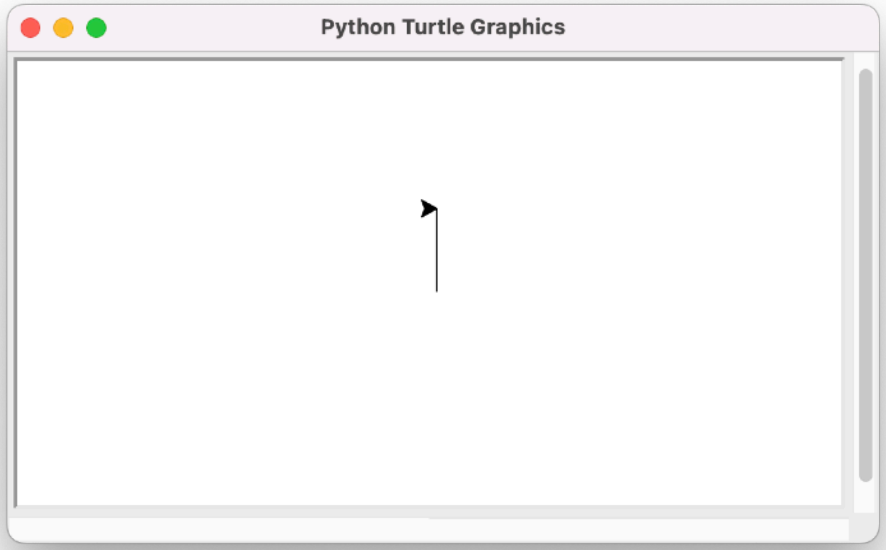


Μετακίνηση στην οθόνη

Με την εντολή setpos() μπορούμε να στείλουμε τη χελώνα μας σε όποιο σημείο της οθόνης θέλουμε. Θα πρέπει να θυμηθούμε πως η setpos() χρειάζεται δύο πληροφορίες (αριθμούς): ο πρώτος αριθμός αφορά τη σειρά στην οποία είμαστε, και ο δεύτερος τη στήλη. Η αρχική θέση είναι πάντα η (0,0).

Αν θέλουμε να "στείλουμε" τη χελώνα μας σε διαφορετική θέση, τότε χρησιμοποιούμε την εντολή:

```
setpos(0,50)
```



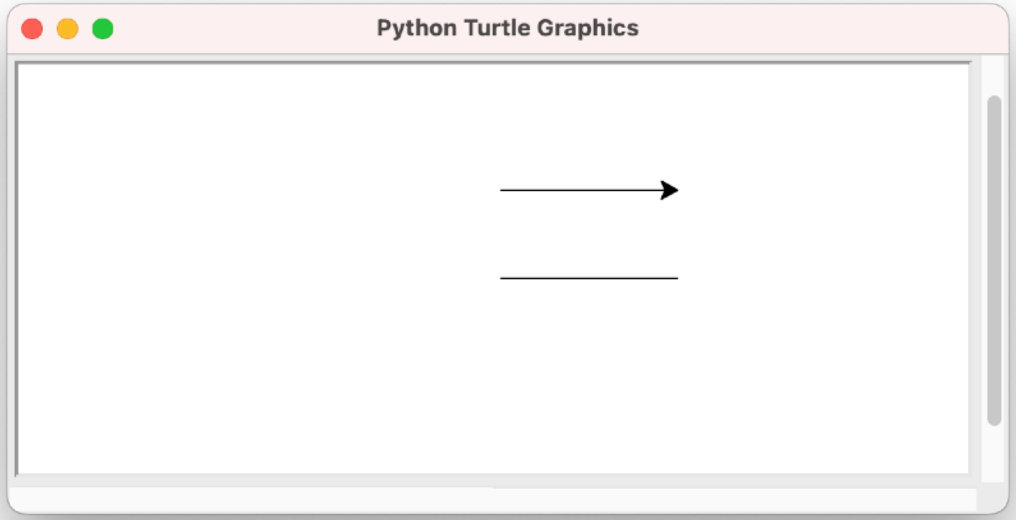
Η χελώνα μας μετακινήθηκε προς τα πάνω 50px, όμως σχεδίασε και γραμμή κατά τη μετακίνηση!

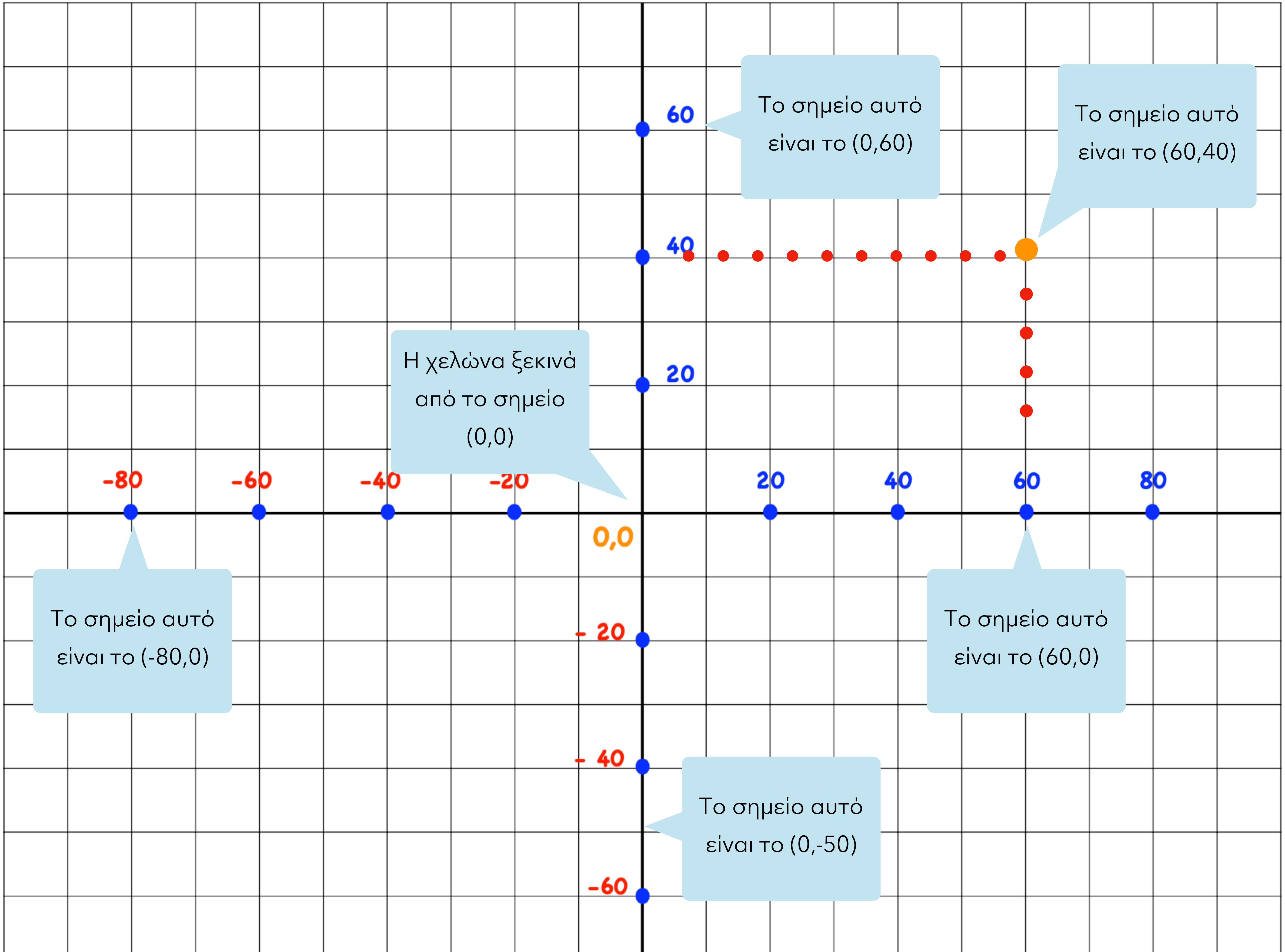
Αν δεν επιθυμούμε να δημιουργηθεί γραμμή κατά τη μετακίνηση, τότε χρησιμοποιούμε και τις εντολές penup() και pendown().



Ας δούμε ένα παράδειγμα:

```
#από τη θέση (0,0) σχεδιάζει γραμμή 100px δεξιά
forward(100)
penup() #η πένα σταματά να γράφει
#η μετακινείται 50px πάνω από την αρχική θέση
setpos(0,50)
pendown() #η πένα αρχίζει να γράφει
#από τη θέση (0,50) σχεδιάζει γραμμή 100px δεξιά
forward(100)
```





Το σημείο αυτό είναι το (0,60)
Το σημείο αυτό είναι το (60,40)
60
40
20
Η χελώνα ξεκινά από το σημείο (0,0)
-80
-60
-40
-20
20
40
60
80
0,0
Το σημείο αυτό είναι το (-80,0)
Το σημείο αυτό είναι το (60,0)
- 20
- 40
-60
Το σημείο αυτό είναι το (0,-50)

Επαναληπτικά σχήματα!

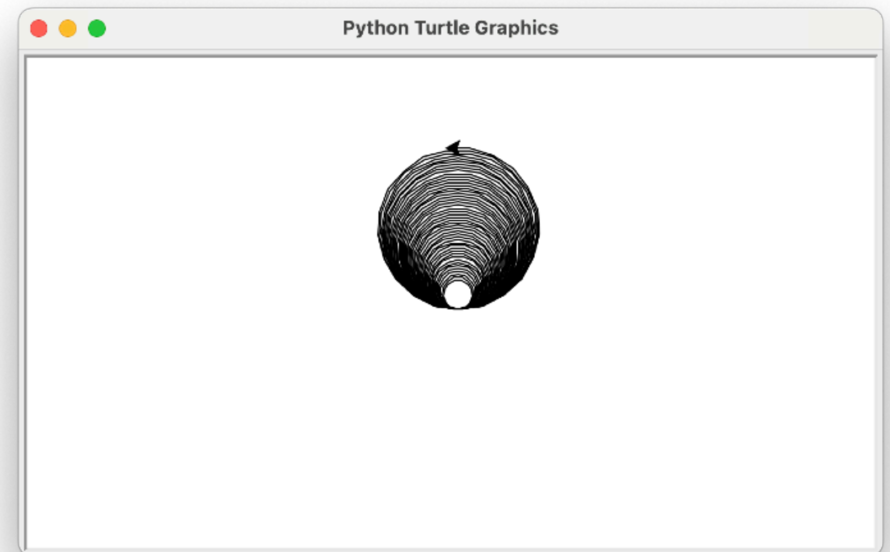
Σε προηγούμενη σελίδα, χρησιμοποιήσαμε επανάληψη για να δημιουργήσουμε τετράπλευρα. Στα συγκεκριμένα παραδείγματα, απλοποιήσαμε τις εντολές (μειώσαμε κατά 6 τις εντολές δημιουργίας τετραγώνου).

Στη συνέχεια θα χρησιμοποιήσουμε επαναλήψεις για να δημιουργήσουμε αριθμό σχημάτων στα οποία θα αλλάζουν οι διαστάσεις αυτόματα.

```
from turtle import *
for x in range(10,110):
    circle(x)
```

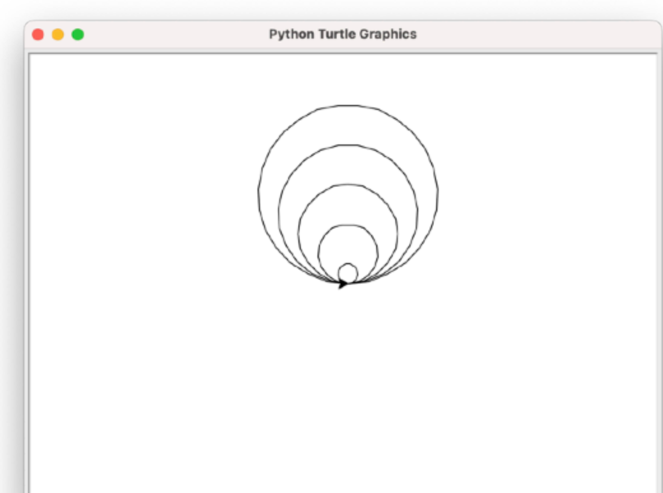


Με την επανάληψη αυτή, θα δημιουργηθούν κύκλοι με ακτίνα από 10 μέχρι 110 pixels, σύμφωνα και με την range().Η επανάληψη συνεχίζεται, με τιμές που παίρνει η μεταβλητή x (από 10 μέχρι 110). Όμως, κάθε φορά αυξάνεται μόλις κατά 1 η ακτίνα.



Θα δώσουμε στην range ακόμη μια παράμετρο. Ο πρώτος αριθμός (το 10) δείχνει με ποια ακτίνα να ξεκινήσει. Ο δεύτερος αριθμός (110) δείχνει πού να σταματήσει. Με τον τρίτο αριθμό (20) δείχνει πόσο να αυξάνεται κάθε φορά η ακτίνα.

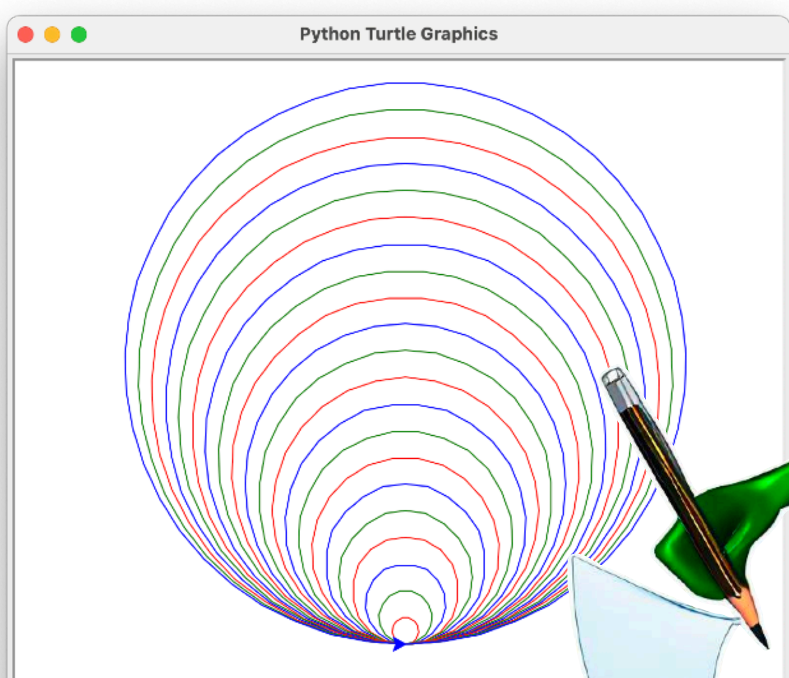
```
from turtle import *
for x in
range(10,110,20):
    circle(x)
```



Επαναλήψεις & χρώματα

Έχουμε δημιουργήσει μια σειρά από κύκλους, οι οποίοι ξεκινούν από το ίδιο σημείο (προηγούμενη σελίδα). Έχουν όμως όλοι το ίδιο χρώμα. Θα χρησιμοποιήσουμε φωλιασμένος επαναλήψεις για να αλλάζουμε κάθε φορά το χρώμα κάθε κύκλου!

```
from turtle import *
x=10 #αρχική τιμή της ακτίνας
while x<200:
    for y in ("red","green","blue"):
        circle(x) #σχηματίζει κύκλο ακτίνας x
        color(y) #αλλάζει κάθε φορά το χρώμα
        x=x+10 #αυξάνεται η ακτίνα κατά 10
```





Στο παράδειγμά μας έχουμε επανάληψη μέσα σε επανάληψη. Με την for y in ("red","green","blue"): επαναλαμβάνουμε 3 φορές τις εντολές που περιλαμβάνει, μέχρι να χρησιμοποιηθούν και τα τρία χρώματα της παρένθεσης.

Με την εντολή **circle(x)** θα δημιουργηθεί ένας κύκλος με ακτίνα x (στη δεύτερη γραμμή, θέσαμε ως αρχική ακτίνα το 10). Με την εντολή **color(y)** αλλάζει το χρώμα κάθε φορά που γίνεται η επανάληψη (red, μετά green, μετά blue).

Απαραίτητη είναι η εντολή **x=x+10** ώστε να αλλάζει σε κάθε επανάληψη η ακτίνα, διαφορετικά ο κάθε κύκλος θα σχηματίζεται πάνω στον προηγούμενο!

Η επανάληψη συνεχίζεται μέχρι το x να γίνει 200 (while<200).

Μπορούμε να πειραματιστούμε με τις εντολές και να δούμε πώς αλλάζει το σχήμα. Για παράδειγμα, μπορούμε να αλλάξουμε την τελευταία γραμμή σε x=x+30. Επίσης, μπορούμε να προσθέσουμε χρώματα (yellow, orange, brown, pink κ.α.).

Ας γεμίσουμε χρώμα!

Μέχρι τώρα, τα σχήματα που δημιουργήσαμε ήταν άχρωμα στο εσωτερικό (ή, καλύτερα, λευκά). Μπορούμε να δώσουμε χρώμα στο εσωτερικό ενός σχήματος, αφού πρώτα -όπως είναι λογικό- το δημιουργήσουμε.

```
from turtle import *
circle(50)
```

Με τις πιο πάνω εντολές, θα δημιουργηθεί ένας κύκλος με ακτίνα 50 pixels και λευκό χρώμα γεμίσματος.

Για να γεμίσουμε ένα σχήμα, θα πρέπει πρώτα να επιλέξουμε το χρώμα που θα χρησιμοποιήσουμε. Αυτό μπορεί να γίνει με την εντολή fillcolor() και το χρώμα στην παρένθεση:

```
fillcolor('red')
```

Τα χρώματα μπορούν να επιλεγούν από μια μεγάλη σειρά αποχρώσεων. Στον οδηγό αυτό θα χρησιμοποιούμε τα βασικά χρώματα (red, blue, yellow, green κτλ) και όχι τις αποχρώσεις τους.

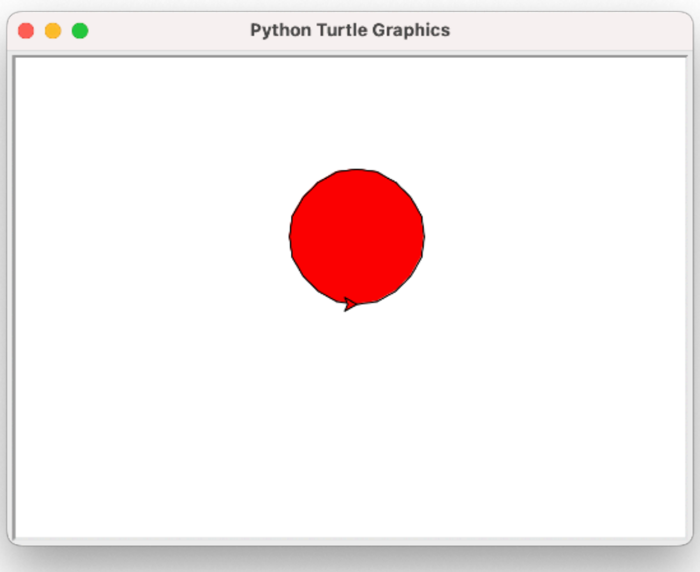


Επειδή υπάρχει περίπτωση να έχουμε αρκετά σχήματα σε ένα πρόγραμμα, με διαφορετικά χρώματα, είναι πολύ σημαντικό να "πούμε" στην Python, από ποιο σχήμα να ξεκινήσει το γέμισμα, και σε ποιο σημείο να σταματήσει.

```
from turtle import *
fillcolor('red') #επιλέγουμε χρώμα
begin_fill() #από αυτό το σημείο ξεκινά το γέμισμα

circle(50)

end_fill() #εδώ σταματά το γέμισμα
```



Σχήματα & Συναρτήσεις

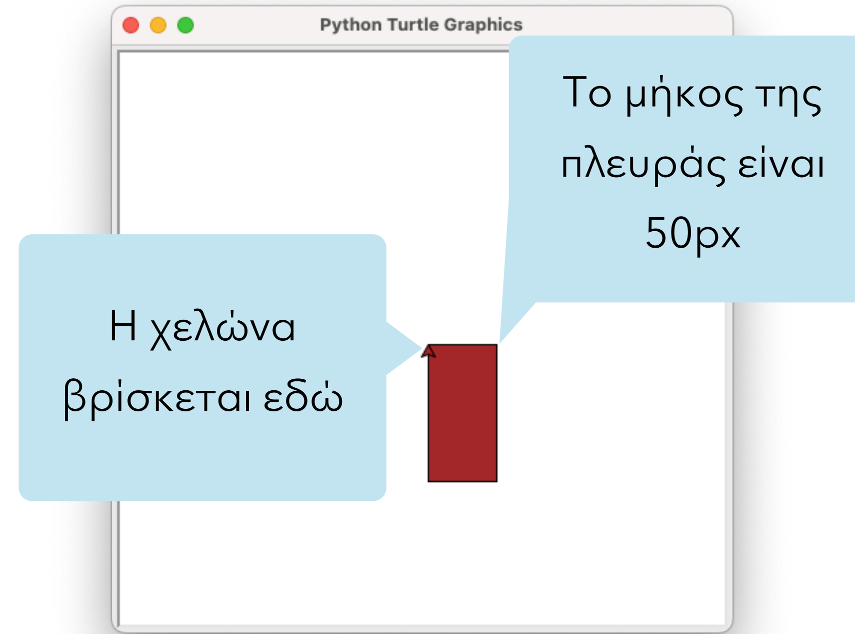
Όταν κατανοήσουμε το πώς κινείται η χελώνα στην οθόνη, μπορούμε να δημιουργήσουμε πολύπλοκα σχήματα. Ας δούμε πώς θα μπορούσαμε να δημιουργήσουμε ένα δέντρο: για τον κορμό του θα σχεδιάσουμε ένα ορθογώνιο, και για το φύλλωμα, έναν κύκλο.

```
from turtle import *

fillcolor("brown")#επιλέγουμε καφέ χρώμα
begin_fill() #ξεκινά το γέμισμα
for x in range(0,2):
    forward(50)
    right(90)
    forward(100)
    right(90)
end_fill()#ολοκληρώνεται το γέμισμα
```

Το ορθογώνιο που δημιουργήσαμε είναι ο κορμός του δέντρου, έτσι έχουμε προσθέσει καφέ χρώμα στο εσωτερικό του. Στη συνέχεια θα δημιουργήσουμε και το πάνω μέρος του δέντρου (κύκλος με πράσινο γέμισμα).

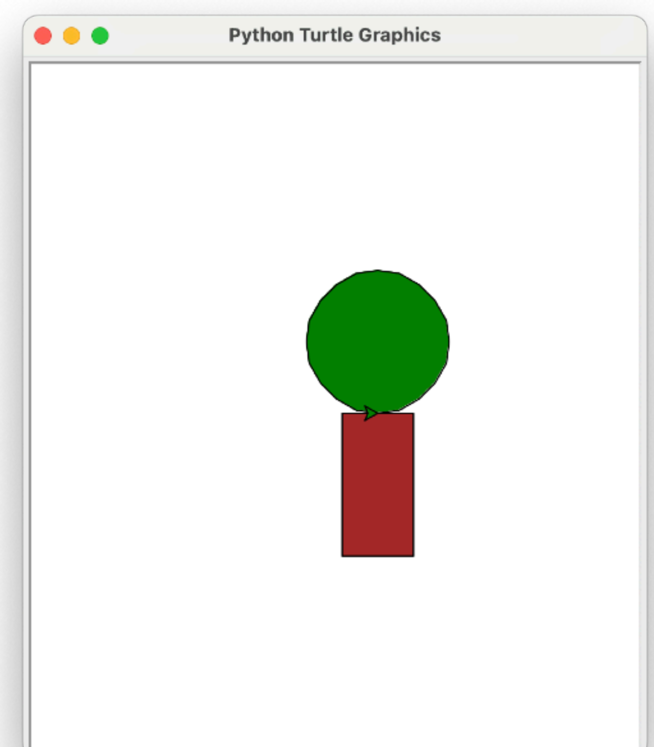
Το φύλλωμα θέλουμε να εμφανίζεται από το μέσο της πάνω πλευράς του ορθογωνίου (εικόνα πάνω δεξιά).



Δίνουμε την εντολή "forward 25" για να πάει στη μέση της γραμμής, και στη συνέχεια δημιουργούμε τον κύκλο (με γέμισμα το πράσινο χρώμα):

```
forward(25)
fillcolor("green")
begin_fill()
circle(50) #το μέγεθος
του φυλλώματος
end_fill()
```

Στη συνέχεια θα δημιουργήσουμε μια σειρά από δέντρα, με τη χρήση συνάρτησης!



Όλος ο κώδικας που είδαμε στην προηγούμενη σελίδα, μας επιτρέπει να δημιουργήσουμε ένα δέντρο. Τον κώδικα θα τον ενσωματώσουμε σε μια συνάρτηση, ώστε να την καλούμε κάθε φορά που θέλουμε να "φυτέψουμε" ένα δέντρο.

```
from turtle import *

def mytree()
    #εντολές για δημιουργία του κορμού
    fillcolor("brown")#επιλέγουμε καφέ χρώμα
    begin_fill() #ξεκινά το γέμισμα
    for x in range(0,2):
        forward(50)
        right(90)
        forward(100)
        right(90)
    end_fill()#ολοκληρώνεται το γέμισμα

    #εντολές για δημιουργία του φυλλώματος
    forward(25)
    fillcolor("green")
    begin_fill()
    circle(50) #το μέγεθος του φυλλώματος
    end_fill()
```

Όλες οι πιο πάνω είναι οι εντολές που (πλέον) περιέχονται στη συνάρτηση mytree().

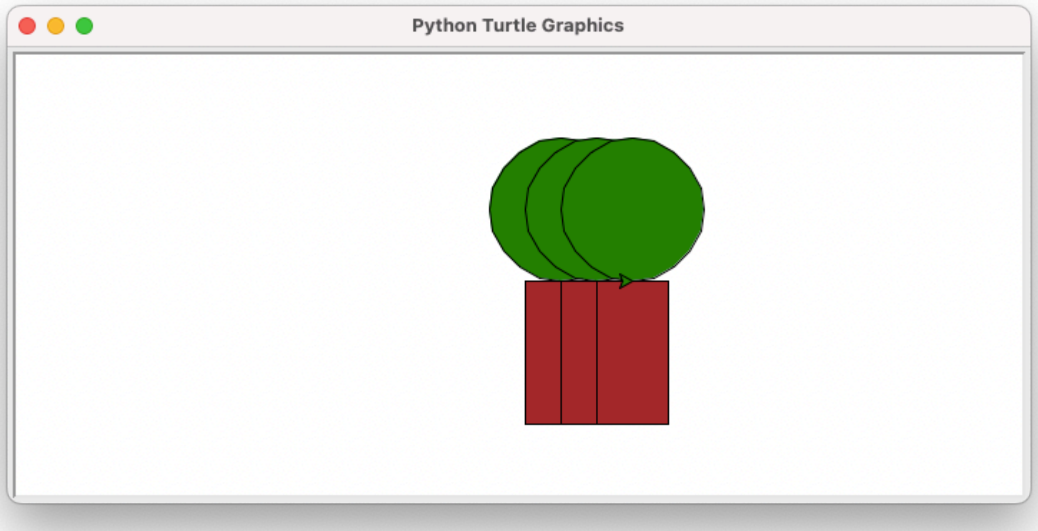


Στη συνέχεια, θα "φυτέψουμε" με κώδικα μια σειρά από δέντρα στην οθόνη μας!

Προσθέτουμε τις εντολές:

```
for x in range(0,3)
    mytree()
```

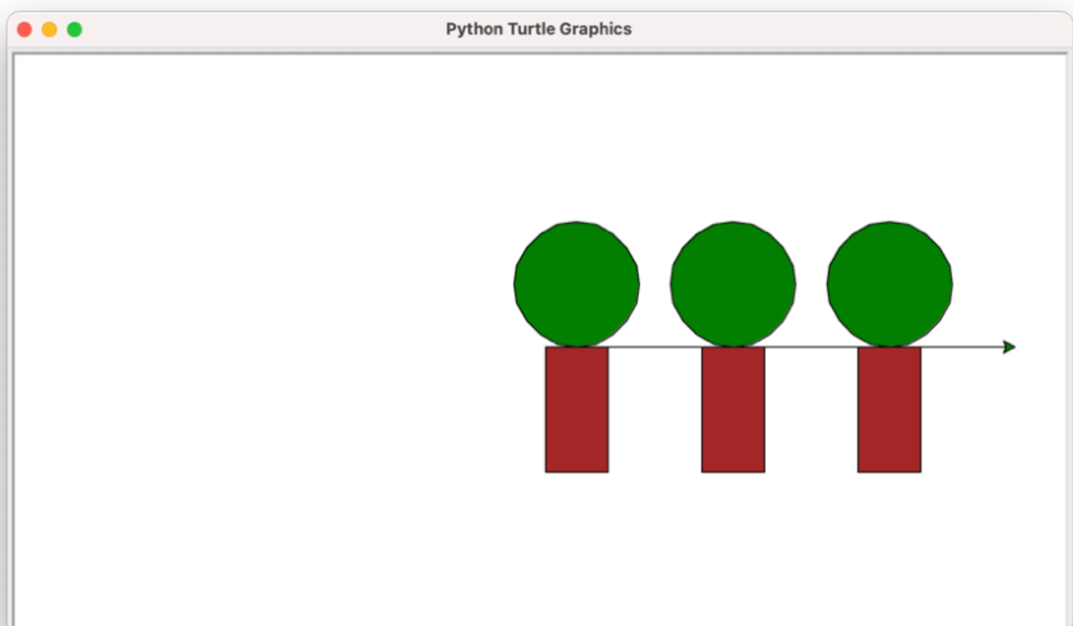
Με τις πιο πάνω εντολές, καλούμε 3 φορές τη συνάρτηση mytree() για να δημιουργηθούν 3 δέντρα. Αν τροποποιήσουμε το range(0,3) και αντί για 3 βάλουμε 5, τότε θα επαναληφθεί η συνάρτηση 5 φορές.



Τα δεντράκια έχουν δημιουργηθεί, αλλά είναι το ένα κολλημένο στο άλλο! Θα πρέπει να κάνουμε ακόμη μια αλλαγή στον κώδικα μας, ώστε να εμφανίζονται το ένα σε απόσταση από το άλλο!

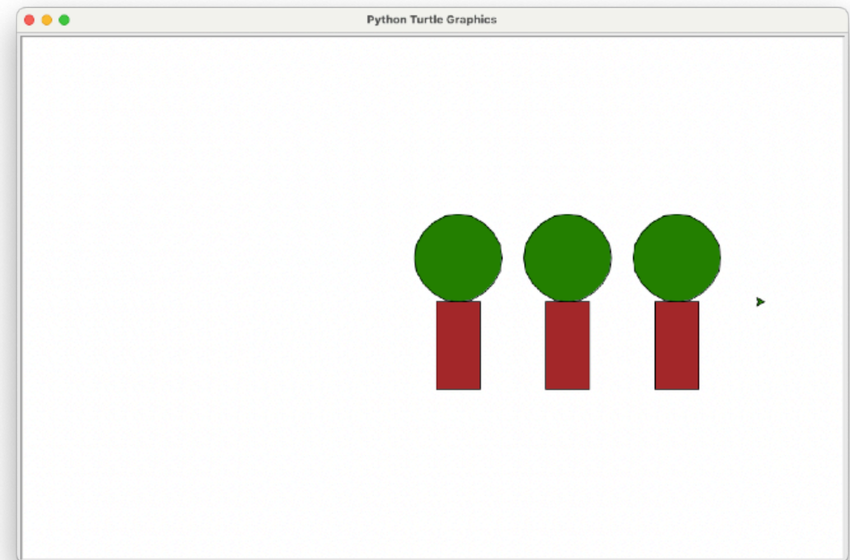
Η συνάρτηση που δημιουργήσαμε μας βοήθησε να σχεδιάσουμε 3 δέντρα στη σειρά (προηγούμενη σελίδα). Όμως, αυτά είναι ενωμένα μεταξύ τους. Θα χρησιμοποιήσουμε την εντολή forward(100) ώστε να κρατήσουμε απόσταση (μπορείτε να αλλάξετε το μέγεθος όπως προτιμάτε).

```
for x in range(0,3)
    mytree() #καλούμε τη συνάρτηση
    forward(100) #κρατάμε απόσταση 100px
```



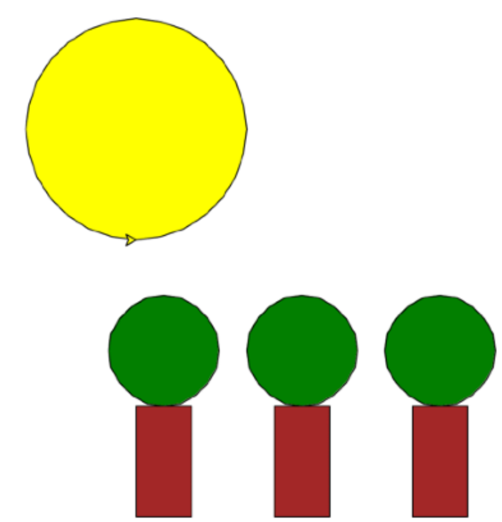
Έχουμε κρατήσει απόσταση μεταξύ των δέντρων, όμως στη μετακίνηση της χελώνας έχει τραβήξει και μια γραμμή από το ένα στο άλλο! Διορθώνεται εύκολα με τις εντολές penup() και pendown().

```
for x in range(0,3)
    mytree() #καλούμε τη συνάρτηση
    penup()
    forward(100) #κρατάμε απόσταση 100px
    pendown()
```



Τα δέντρα μας εμφανίζονται με τη σειρά! Αν θέλουμε να πειραματιστούμε, μπορούμε να προσθέσουμε και ήλιο!

```
penup()
setpos(0,150)
pendown
fillcolor("yellow")
begin_fill()
circle(50)
end_fill()
```



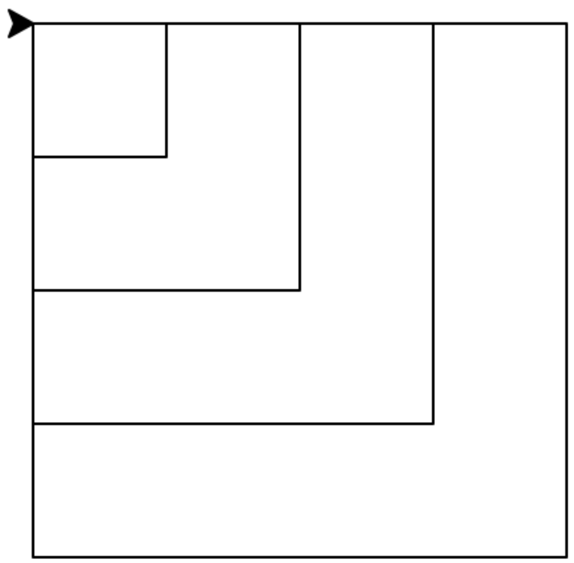
Τι μάθαμε μέχρι τώρα

- Στο **Μέρος Γ' "Χελώνες και Πύθωνες"** εργαστήκαμε με πρόσθετες εντολές δημιουργίας γεωμετρικών σχημάτων!
- Εισηγάγαμε και χρησιμοποιήσαμε το άρθρωμα "turtle" που επιτρέπει στην Python να δημιουργήσει γεωμετρικά σχήματα.
- Γνωρίσαμε τον τρόπο με τον οποίο κινείται η χελώνα μας στην οθόνη, με σημείο έναρξης τη θέση (0,0) (θέση "home").
- Χρησιμοποιήσαμε εντολές κίνησης (forward, left, right κτλ) για να μετακινήσουμε τη χελώνα και να κατασκευάσει σχήματα.
- Αξιοποιήσαμε τις επαναλήψεις για να απλοποιήσουμε τον κώδικα μας.
- Δημιουργήσαμε συνθήκη για κατασκευή πολύπλοκων σχημάτων.



Δραστηριότητες

1. Να γράψετε κώδικα με τον οποίο να δημιουργήσετε 4 τετράγωνα διαφορετικού μεγέθους, το ένα μέσα στο άλλο, όπως στο σχήμα:



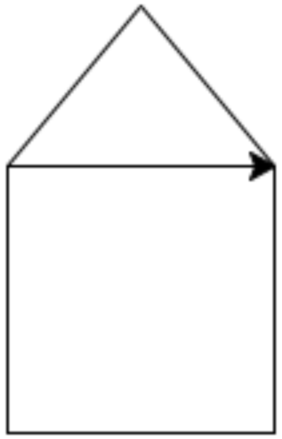
2. Να αλλάξετε τον πιο κάτω κώδικα, ώστε ο κάθε κύκλος να έχει διαφορετικό χρώμα:

```
from turtle import *
x=10 #αρχική τιμή της ακτίνας
while x<200:
 for y in (“red”,”green”,”blue”):
     circle(x) #σχηματίζει κύκλο ακτίνας x
     color(y) #αλλάζει κάθε φορά το χρώμα
     x=x+10 #αυξάνεται η ακτίνα κατά 10
```

3. Να σχεδιάσετε το σχήμα που θα εμφανιστεί όταν εκτελέσετε το πιο κάτω πρόγραμμα:

```
from turtle import *
y=20
for k in range(0,4):
  for x in range(0,4):
        color(“blue”)
        circle(y)
  y=y+20
```

4. Να γράψετε τον κώδικα με τον οποίο σχηματίζεται ένα σπιτάκι, όπως φαίνεται στην εικόνα πιο κάτω:



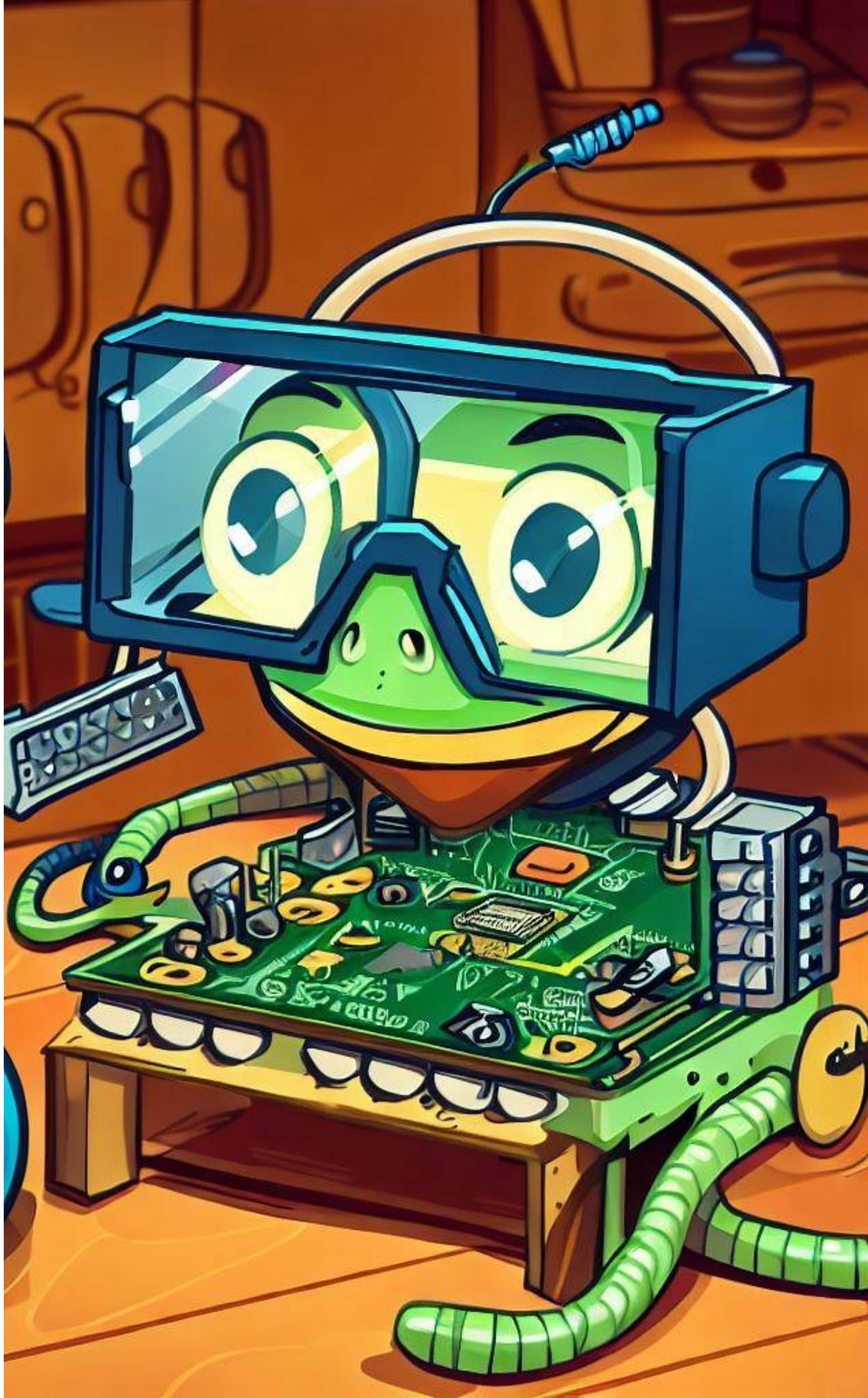
ΜΕΡΟΣ Δ΄: ΡΟΜΠΟΠΥΘΩΝΕΣ!



IoT, Ρομπότ & Python

Στα προϊστορικά χρόνια, τότε που ο άνθρωπος ζούσε σε σπηλιές και κυνηγούσε με το ρόπαλο, οι υπολογιστές ήταν μεγάλοι σε μέγεθος και δεν είχαν καν σύνδεση στο διαδίκτυο!

Οκ… δεν ήταν ΤΟΣΟ πίσω οι υπολογιστές που δεν ήταν συνδεδεμένοι στο διαδίκτυο, αλλά από το τέλος του 1970 μέχρι και τις αρχές του 2000, οι υπολογιστές καταλάμβαναν το μεγαλύτερο μέρος του γραφείου μας, ενώ οι φορητοί υπολογιστές ήταν πανάκριβοι. Από τη δεύτερη δεκαετία του 2000 και μετά, άλλαξαν πολλά. Οι υπολογιστές πλέον έγιναν πολύ μικροί, ένα κινητό τηλέφωνο απέκτησε τη δύναμη που είχαν ταχύτατοι υπολογιστές λίγων μόνο ετών, ενώ αρκετές εταιρείες και οργανισμοί δημιούργησαν ακόμη μικρότερα συστήματα για την εκπαίδευση και τη βιομηχανία. Το Raspberry Pi είναι ένα παράδειγμα τέτοιου υπολογιστή, ενώ τα Arduino έδωσαν μια λύση για ανάπτυξη έξυπνων συσκευών πολύ χαμηλού κόστους!



BBC Micro:bit

```
10 |  print ("Curiouser and curiouser")
20 | #Alice in Wonderland
```

BBC micro:bit

Το γνωρίζατε ότι οι πύθωνες είναι πολύ καλοί ΚΑΙ στα ηλεκτρονικά; Αυτό είναι κάτι που κρατούν ως μεγάλο μυστικό (οκτασφράγιστο, για να σπάει δυσκολότερα) διαφορετικά όλοι στο ζωικό βασίλειο θα τους έπαιρναν τις συσκευές τους για επιδιόρθωση!

Στην ενότητα αυτή θα εργαστούμε με τον προγραμματισμό των εκπληκτικά μικρών και εκπληκτικά εύκολων στον προγραμματισμό, BBC micro:bit. Οι συσκευές αυτές είναι μικροσκοπικοί υπολογιστές και σχεδιάστηκαν ειδικά για την εκπαίδευση, στο Ηνωμένο Βασίλειο αρχικά, και δόθηκαν σε εκατομμύρια παιδιά. Με τη βοήθεια των micro:bits, τα παιδιά μπορούν να μάθουν προγραμματισμό, αλλά και κατασκευή και έλεγχο ρομπότ και άλλων συσκευών.

Έχουν πολλούς αισθητήρες (υγρασίας, θερμοκρασίας κ.α.) ενώ στο μπροστινό τους τμήμα έχουν και ένα πλέγμα με λαμπάκια LED (5 σειρές και 5 στήλες) το οποίο μπορούμε να προγραμματίσουμε εύκολα.

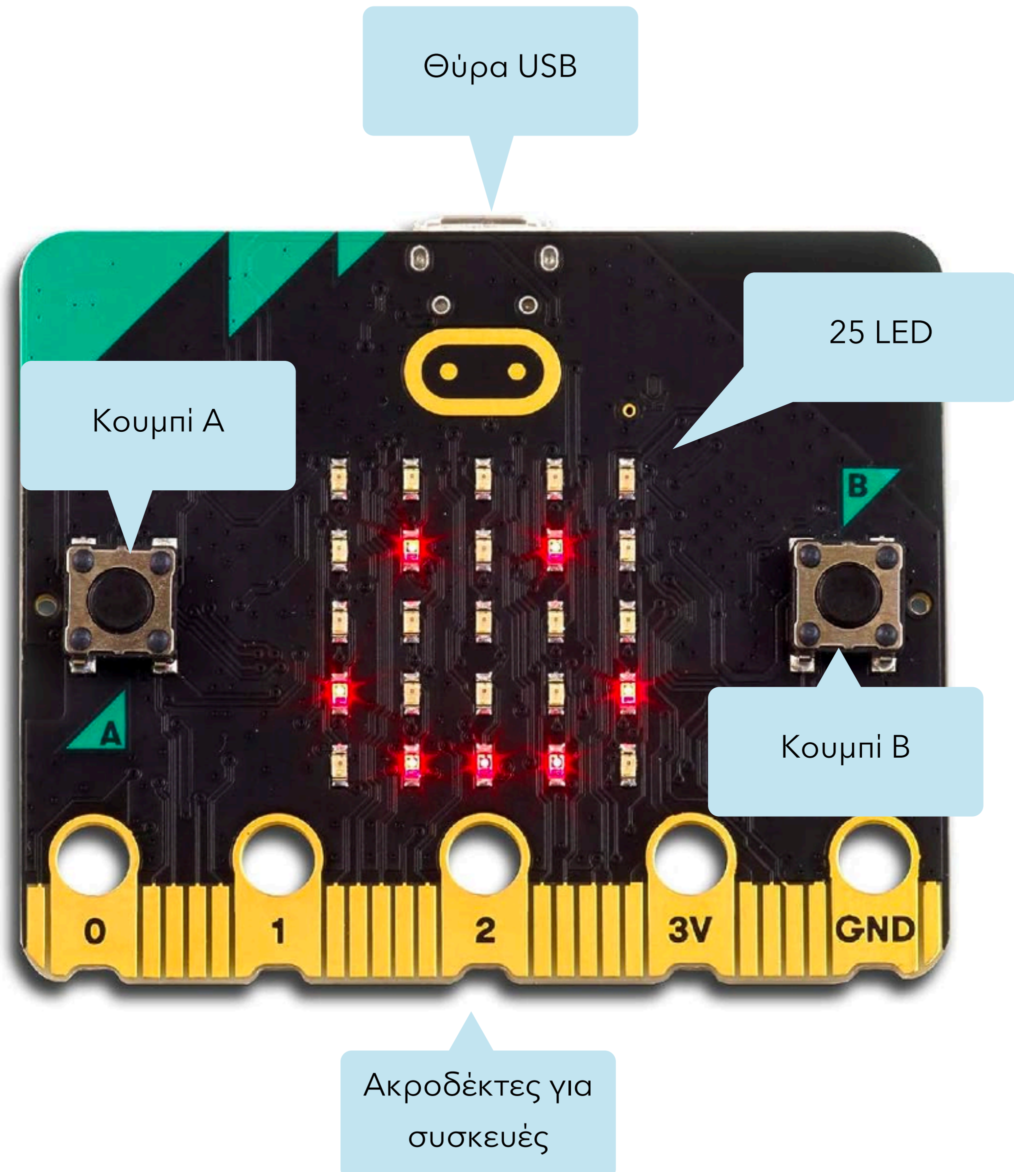


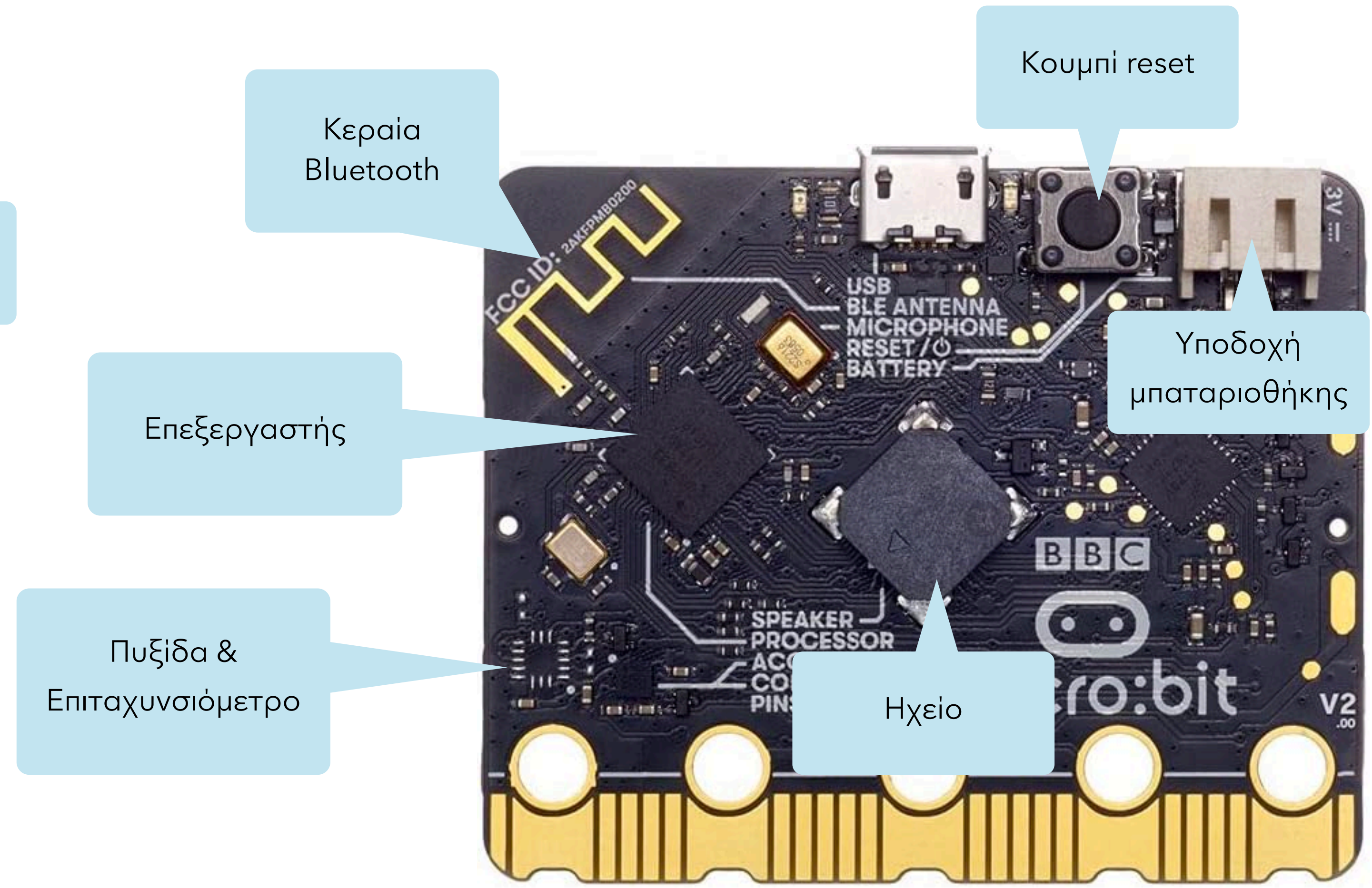
Τι θα γνωρίσουμε:

Μέσα από το **Μέρος Δ' "BBC micro:bit"** θα έχουμε την ευκαιρία:

- Να γνωρίσουμε τον μικροσκοπικό υπολογιστή BBC micro:bit
- Να προγραμματίσουμε το micro:bit με τη βοήθεια του Python Editor και να προβάλουμε πληροφορίες στα LED που βρίσκονται στο μπροστινό μέρος
- Να προγραμματίσουμε τους αισθητήρες του micro:bit (θερμοκρασία, υγρασίας)
- Να προγραμματίσουμε την πυξίδα του micro:bit
- Να δημιουργήσουμε ένα απλό παιχνίδι με τη βοήθεια του micro:bit
- Να γνωρίσουμε το περιβάλλον του MakeCode, ένα εναλλακτικό τρόπο προγραμματισμού του micro:bit







Γνωριμία με το micro:bit

Το micro:bit είναι μια εξαιρετική συσκευή! Μας επιτρέπει να το προγραμματίσουμε, να παίξουμε παιχνίδια, να κατασκευάσουμε και να προγραμματίσουμε ρομπότ, να το χρησιμοποιήσουμε σε επιστημονικές μετρήσεις και άλλα πολλά! Οι δυνατότητες του είναι απίστευτες, ενώ είναι σχετικά φθηνό και πάρα πολύ απλό στη λειτουργία.

Αν και δεν έχει δυνατότητα να συνδεθεί με εξωτερική οθόνη, το micro:bit είναι ένας ολοκληρωμένος υπολογιστής, που μπορεί να προγραμματιστεί και να εκτελέσει ιδιαίτερα πολύπλοκες λειτουργίες.

Στις εικόνες πιο πάνω βλέπουμε τα μέρη του micro:bit (μπροστινό μέρος, πάνω αριστερά, πίσω μέρος, πάνω δεξιά).

BBC Micro Computer

Το BBC, το κρατικό κανάλι του Ηνωμένου Βασιλείου, ξεκίνησε το 1982 την πρώτη τηλεοπτική εκπαιδευτική σειρά για ηλεκτρονικούς υπολογιστές. Μια εταιρεία, η Acorn, κέρδισε το συμβόλαιο για τη δημιουργία ενός υπολογιστή για την εκπαίδευση. Ο υπολογιστής αυτός αρχικά ονομαζόταν Proton, όμως άλλαξαν το όνομα του σε BBC Micro. Η επιτυχία του εκπαιδευτικού τηλεοπτικού προγράμματος ήταν τόσο μεγάλη, που σιγά σιγά όλα σχεδόν τα δημόσια σχολεία του Ηνωμένου Βασιλείου απέκτησαν το BBC Micro. Αρκετοί γονείς αγόρασαν τον υπολογιστή αυτό και για το σπίτι, ώστε να συνεχίσουν να μαθαίνουν τα παιδιά τους και μετά το σχολείο.

Οι δημιουργοί του BBC Micro, πολύ σωστά και σοφά, έβαλαν στο μηχάνημα μια γλώσσα προγραμματισμού, τη BASIC, που ξεκινούσε μαζί με τον υπολογιστή. Πολλά παιδιά στο Ηνωμένο Βασίλειο, αλλά και σε άλλες χώρες -όπως η Κύπρος- έκαναν τα πρώτα τους βήματα στον προγραμματισμό με αυτόν τον υπολογιστή.



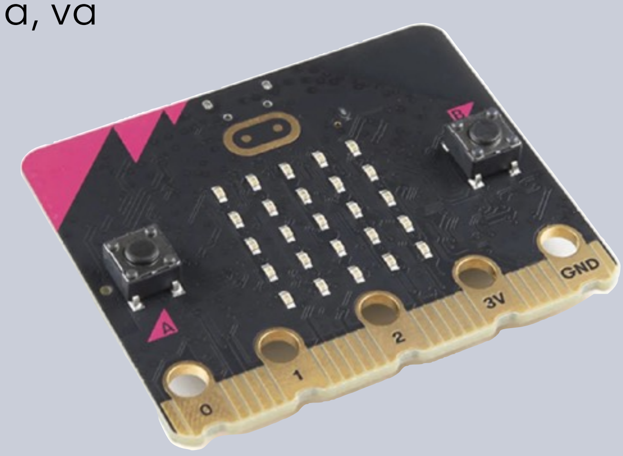
Η Acorn, εταιρεία που κατασκεύασε το BBC Micro, δημιούργησε επίσης και τον επεξεργαστή ARM που αποτελεί τον "εγκέφαλο" των σημερινών tablets, έξυπνων τηλεφώνων, υπολογιστών (π.χ. Apple) αλλά και άλλων συσκευών.

BBC micro:bit

Το 2015, το BBC δημιούργησε ένα νέο εκπαιδευτικό πρόγραμμα με στόχο να προσφέρει σε εκατομμύρια μαθητές στο Ηνωμένο Βασίλειο μια ψηφιακή συσκευή για να μάθουν προγραμματισμό (και όχι μόνο!).

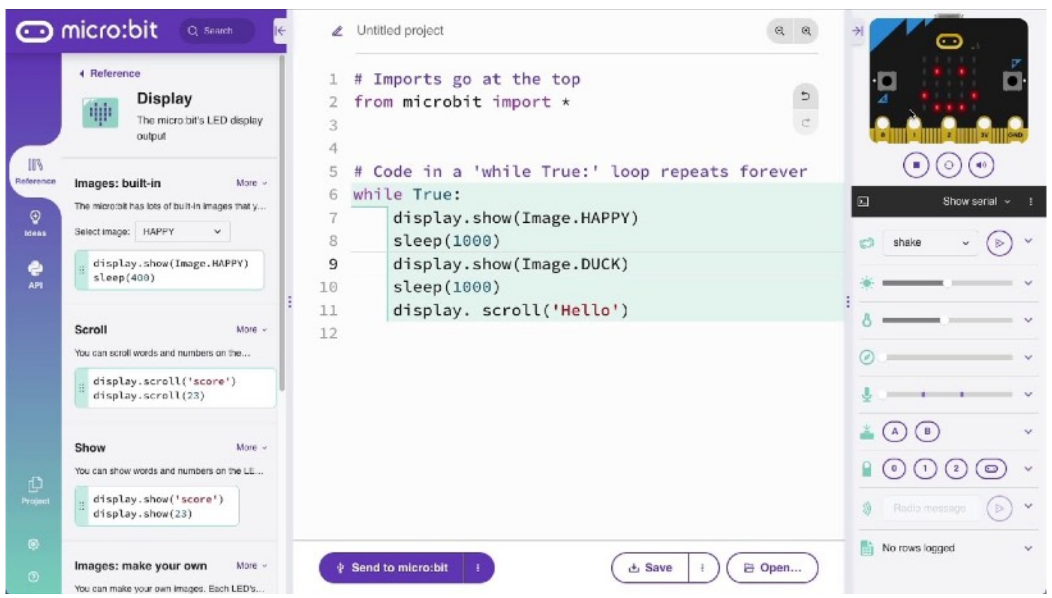
Η συσκευή έχει το μέγεθος πιστωτικής κάρτας και βασίζεται σε επεξεργαστή ARM - τον απόγονο του τελευταίου BBC υπολογιστή με το οποίο ξεκίνησε το 1982!

Το BBC Micro Bit (ή micro:bit όπως παρουσιάζεται) είναι ένας μικρός μεν αλλά αρκετά ικανός υπολογιστής, ο οποίος περιλαμβάνει αρκετούς αισθητήρες που μπορούμε να χρησιμοποιήσουμε για να μάθουμε προγραμματισμό, να δημιουργήσουμε απλά παιχνίδια, να κατασκευάσουμε και να προγραμματίσουμε ρομπότ, να κάνουμε μετρήσεις από το περιβάλλον κ.α.



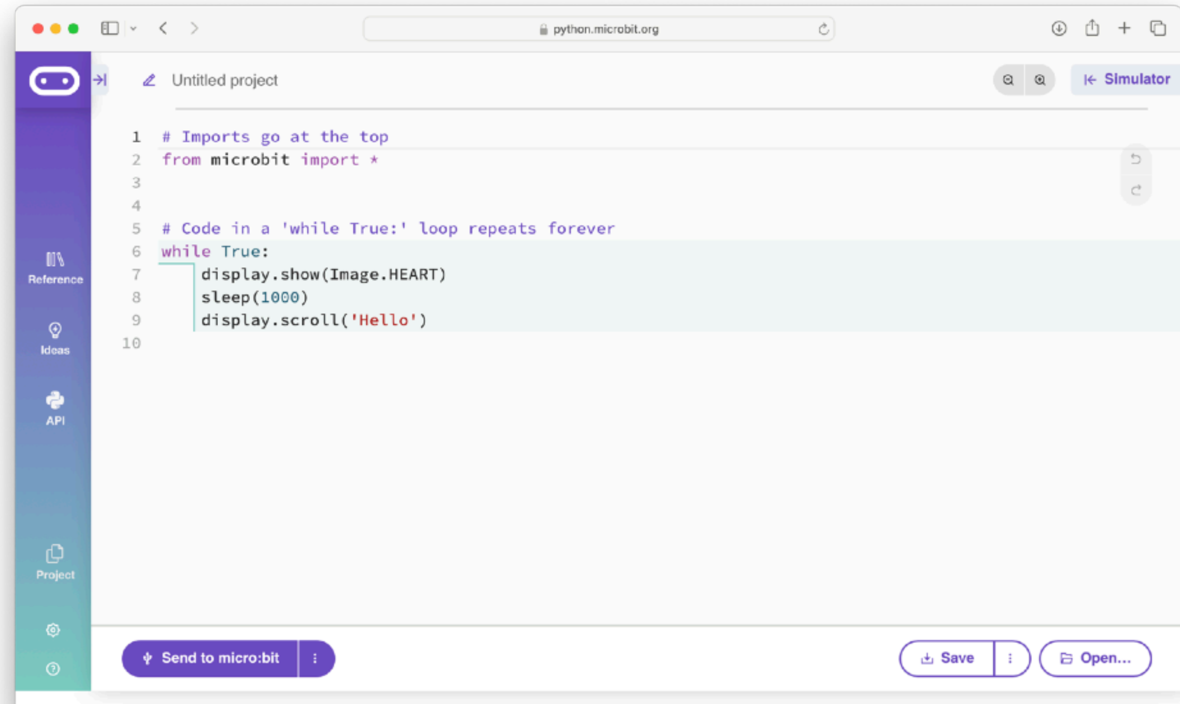
Python Editor

Το micro:bit είναι μια συσκευή που επιτρέπει τον προγραμματισμό του με αρκετά και διαφορετικά περιβάλλοντα. Μπορούμε να αξιοποιήσουμε το Microsoft MakeCode (βλέπε τέλος του κεφαλαίου) ή το Scratch (ανάμεσα σε άλλα). Για τα παραδείγματα που ακολουθούν θα χρησιμοποιήσουμε το Python Editor της σελίδας https://python.microbit.org.



Βίντεο ιστού - κάντε κλικ για αναπαραγωγή

Με την πρώτη επίσκεψη στη σελίδα του Python Editor, προβάλλεται ένα εισαγωγικό βίντεο. Το περιβάλλον είναι ιδιαίτερα απλό στον σχεδιασμό και επιτρέπει τον έλεγχο του κώδικα μέσω προσομοίωσης (simulation). Αυτή η δυνατότητα είναι πολύ χρήσιμη, ειδικά αν δεν έχουμε συσκευές micro:bit.



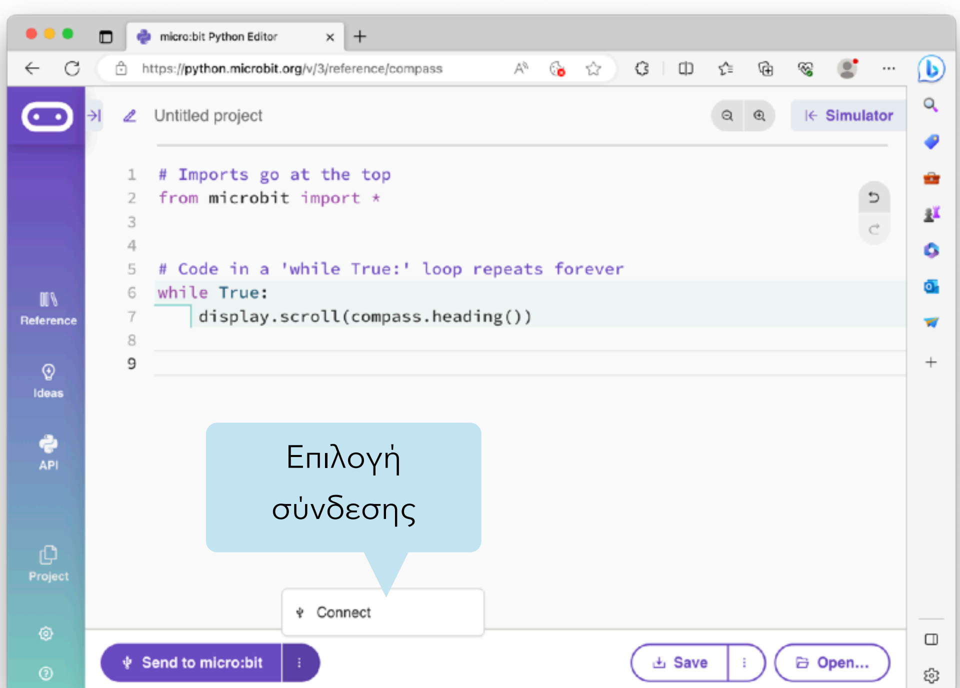
Το περιβάλλον του Python Editor είναι πολύ απλό και μας βοηθά στο να γράψουμε τον κώδικά μας. Μέρη του κώδικα χρωματίζονται ώστε να διευκολυνθούμε στην ανάγνωσή του. Επίσης εμφανίζεται η αρίθμηση των γραμμών του κώδικα, κάτι που μας βοηθά όταν θέλουμε να εντοπίσουμε και να διορθώσουμε σφάλματα.

Στην επόμενη σελίδα θα γνωρίσουμε περισσότερα για το Python Editor.

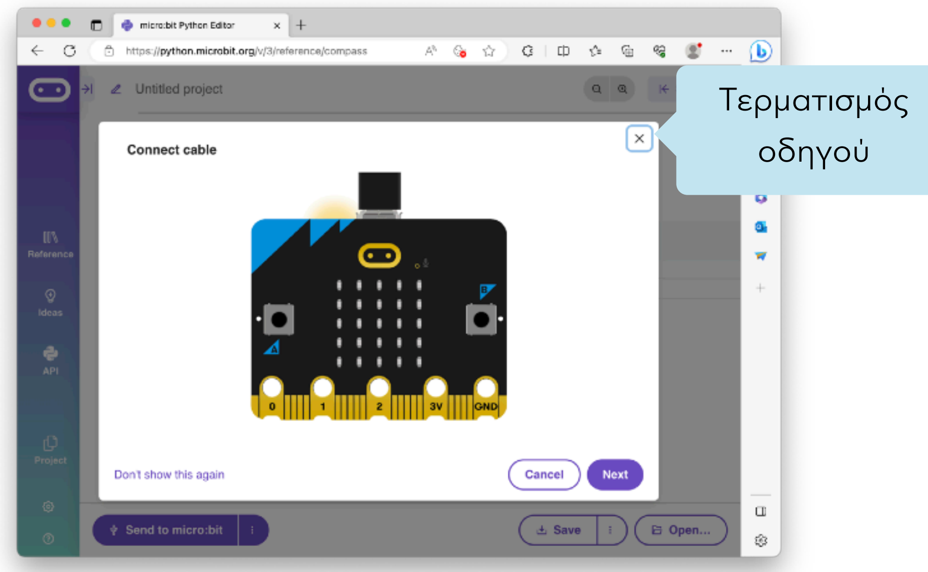
Σύνδεση micro:bit

Ο ιδανικός τρόπος λειτουργίας του Python Editor, είναι σε συνδυασμό με πραγματικό micro:bit. Βέβαια, οι δημιουργοί του Editor δημιούργησαν και ένα εικονικό micro:bit, ώστε να ελέγχουμε τον κώδικά μας ακόμη και αν δεν έχουμε τη συσκευή.

Για σύνδεση του micro:bit, θα πρέπει πρώτα να το ενώσουμε με τον υπολογιστή μας με USB. Στη συνέχεια, κάνουμε κλικ στην αντίστοιχη επιλογή στο κάτω μέρος της οθόνης ώστε να συνδεθεί και το Python Editor με τη συσκευή μας (εικόνα κάτω).

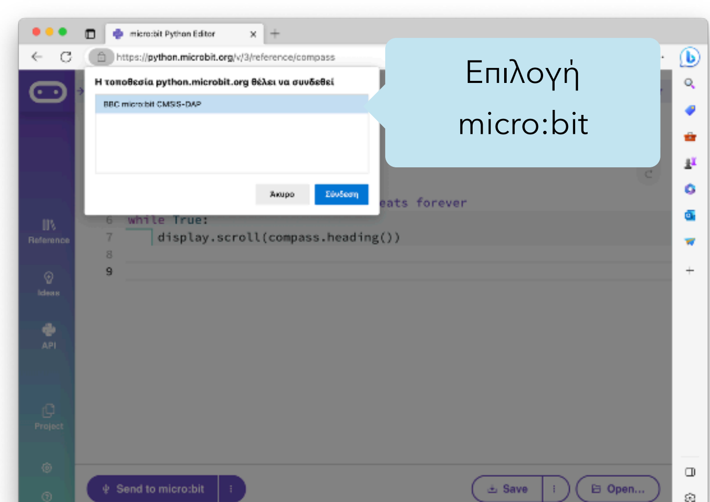


Αφού πατήσουμε για σύνδεση, θα εμφανιστεί ένας σύντομος οδηγός που θα εξηγεί (με εικόνες) το πώς να προχωρήσουμε (εικόνα κάτω).



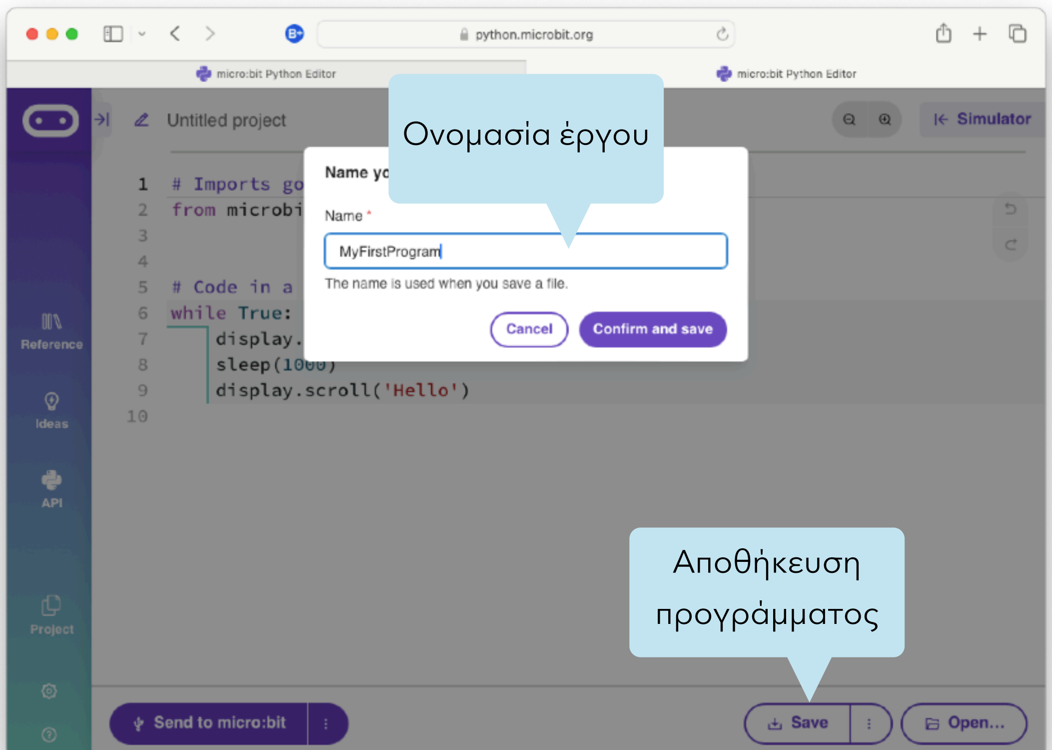
Μπορούμε να ακολουθήσουμε τον οδηγό ή να πατήσουμε στο Χ πάνω δεξιά για να τον κλείσουμε.

Στη συνέχεια, επιλέγουμε το micro:bit μας και πατάμε το κουμπί "Σύνδεση".



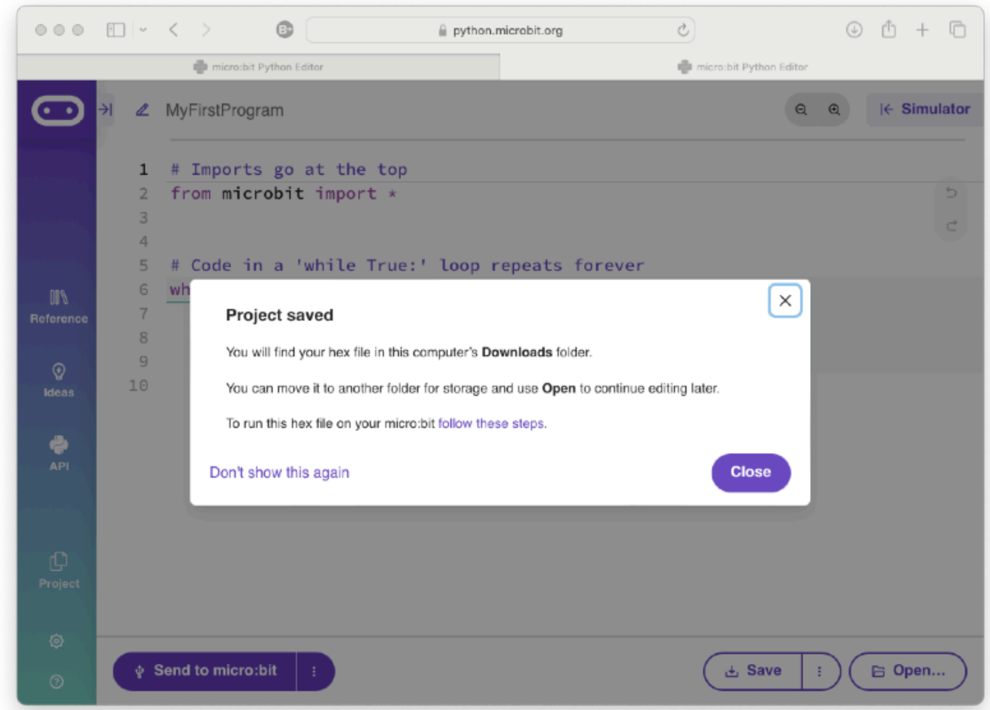
Αποθήκευση αρχείου

Για να εκτελεστεί ένα πρόγραμμα στο micro:bit, αφού το συνδέσουμε με τον υπολογιστή μας (προηγούμενη σελίδα), θα πρέπει να το αποθηκεύσουμε και να το μεταφέρουμε.



Για να αποθηκεύσουμε ένα πρόγραμμα ("Έργο" όπως ονομάζεται) κάνουμε κλικ στο κουμπί "Save" στο κάτω μέρος. Στο πλαίσιο που εμφανίζεται (εικόνα πάνω) πληκτρολογούμε ένα όνομα για το αρχείο και στη συνέχεια το κουμπί "Confirm and save".

Στη συνέχεια θα εμφανιστεί ένα μήνυμα που θα μας ενημερώνει σε ποιο χώρο στον υπολογιστή μας αποθηκεύτηκε το αρχείο.



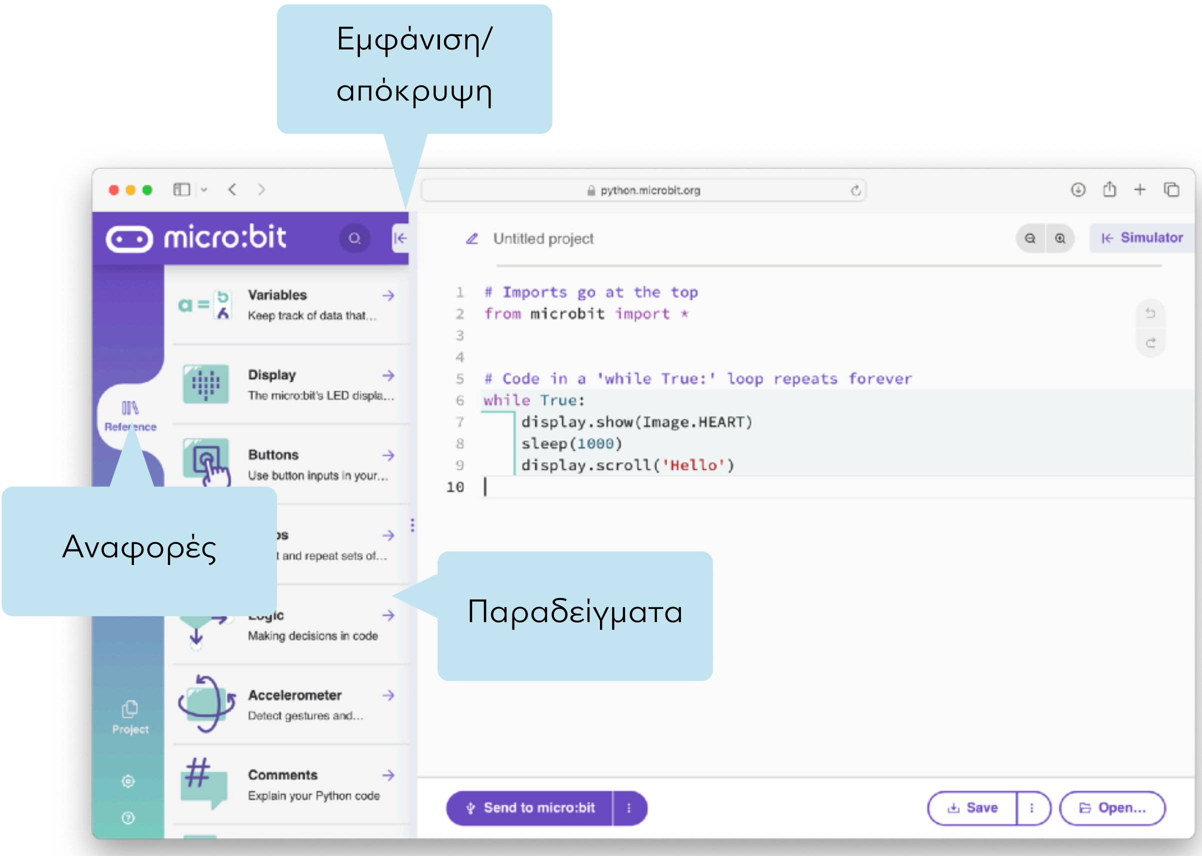
Για να ανοίξουμε ένα αρχείο, κάνουμε κλικ στο κουμπί "Open" στο κάτω μέρος της οθόνης. Στο πλαίσιο που εμφανίζεται, επιλέγουμε το αρχείο που θέλουμε να ανοίξει ώστε να εμφανιστεί στην οθόνη μας.

Όταν προγραμματίζουμε σε Python Editor για micro:bit, δεν έχει νόημα να εκτελέσουμε πρόγραμμα που γράψαμε για λειτουργία σε υπολογιστή με οθόνη και πληκτρολόγιο! Στον Python Editor θα δουλεύουμε μόνο για χρήση με micro:bit!

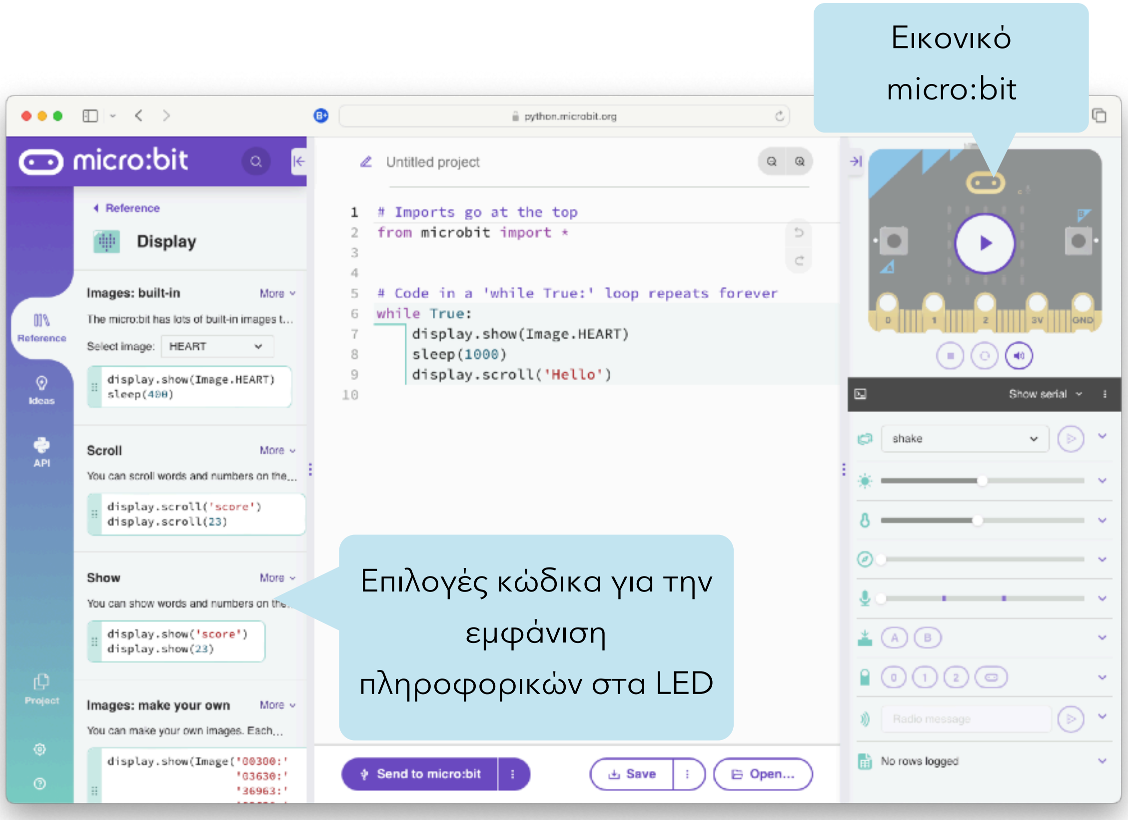


Έτοιμος κώδικας

Για αρχάριους στη χρήση Python, υπάρχει η δυνατότητα εμφάνισης των βασικών εντολών. Κάνουμε κλικ στο “Αναφορές” (εικόνα κάτω) για να εμφανιστούν οι βασικές εντολές.



Οι εντολές είναι πολύ βοηθητικές, καθώς δίνουν και πληροφορίες για τις δυνατότητες του micro:bit. Για να εμφανιστούν περισσότερες πληροφορίες, αλλά και δείγμα κώδικα, κάνουμε κλικ σε μια από τις (πολλές) επιλογές που υπάρχουν (εικόνα πάνω δεξιά).



Το εικονικό micro:bit μας επιτρέπει να ελέγξουμε τον κώδικα μας ακόμη και αν δεν έχουμε τη συσκευή. Στην εικόνα πάνω, στο δεξί τμήμα της, εμφανίζεται το micro:bit. Με κάθε εντολή που δίνουμε, και για κάθε αισθητήρα, κουμπί ή άλλο (π.χ. ηχείο), εμφανίζεται η αντίστοιχη λειτουργία. Κάτω από την εικόνα του micro:bit μπορούμε να δούμε τις ενδείξεις κάθε αισθητήρα, καθώς και τα κουμπιά που υπάρχουν πάνω σ’αυτό. Περισσότερα θα δούμε σε επόμενες σελίδες.

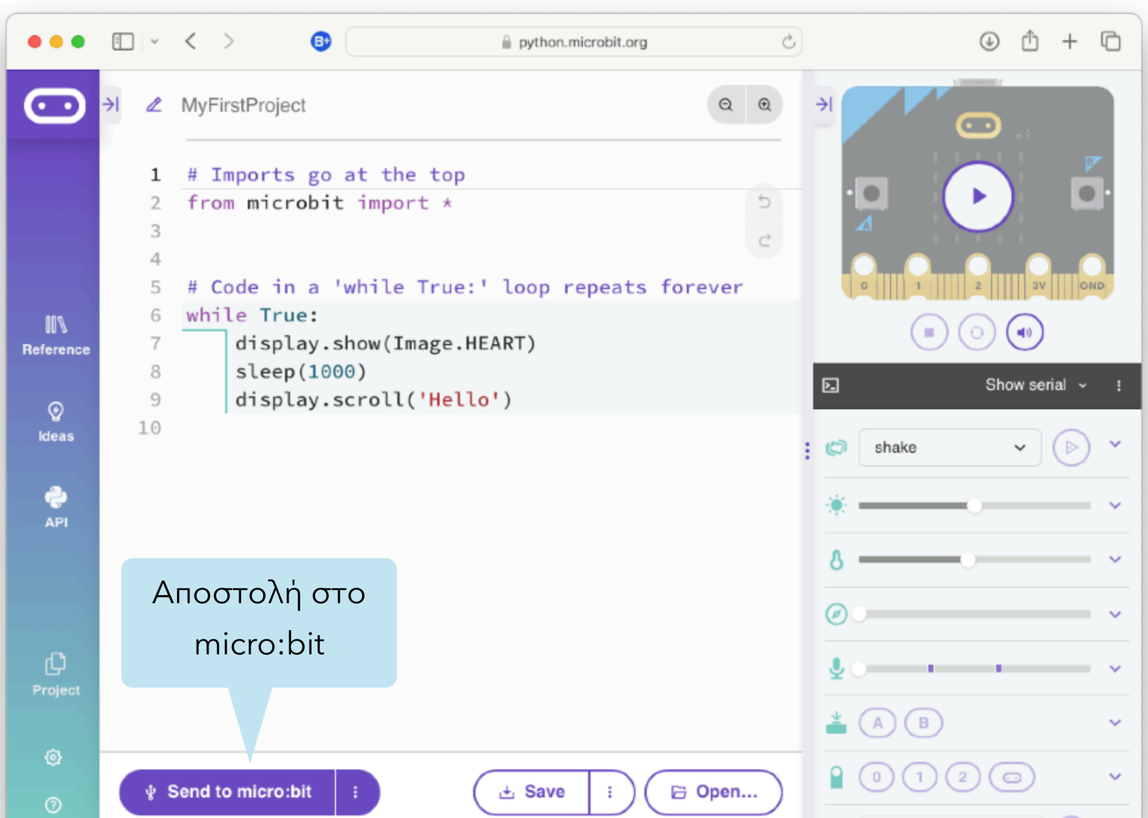


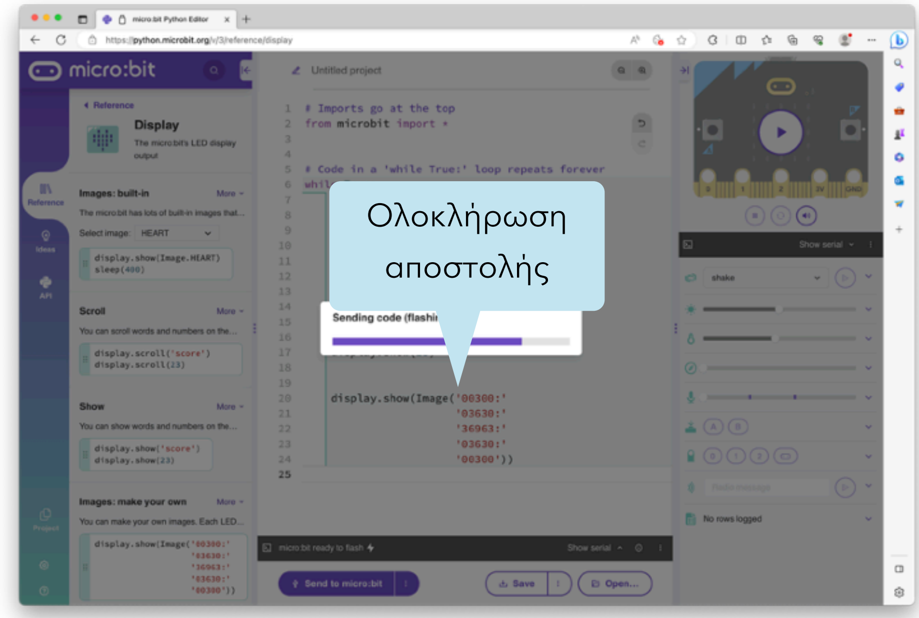


Μεταφορά αρχείου

Αφού γράψουμε τον κώδικα του προγράμματος μας και συνδέσουμε το micro:bit (προηγούμενες σελίδες), θα πρέπει να μεταφέρουμε το πρόγραμμα. Υπάρχουν δύο τρόποι για να γίνει αυτό:

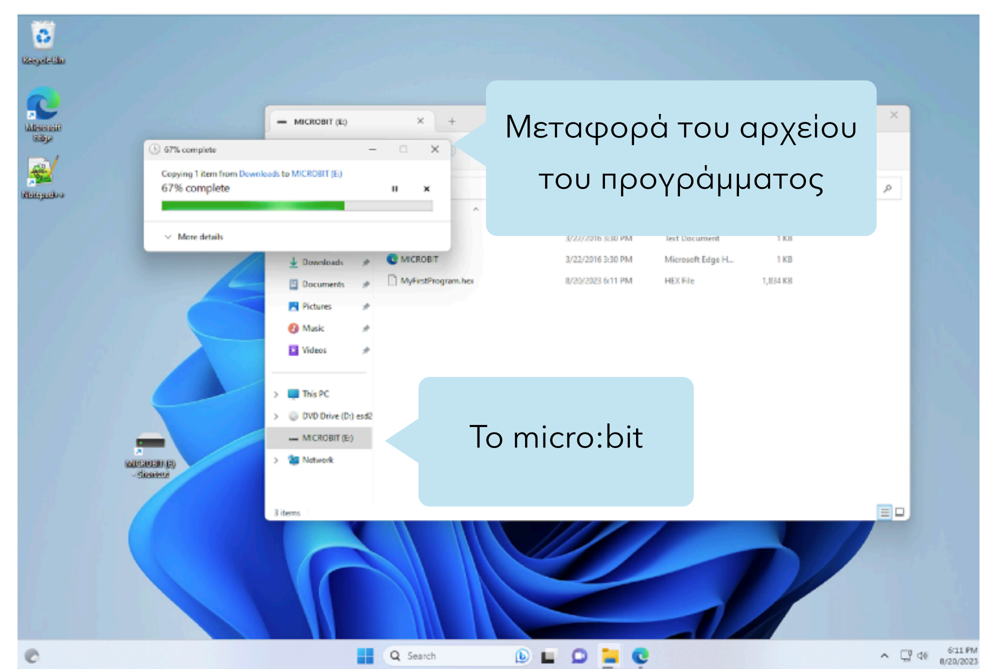
Ο απλός και πιο συνηθισμένος τρόπος είναι απευθείας από τον υπολογιστή μας και το Python Editor: **αφού πρώτα συνδεθεί** το micro:bit, κάνουμε κλικ στο κουμπί "Send to micro:bit" στο κάτω μέρος του παραθύρου (εικόνα κάτω).





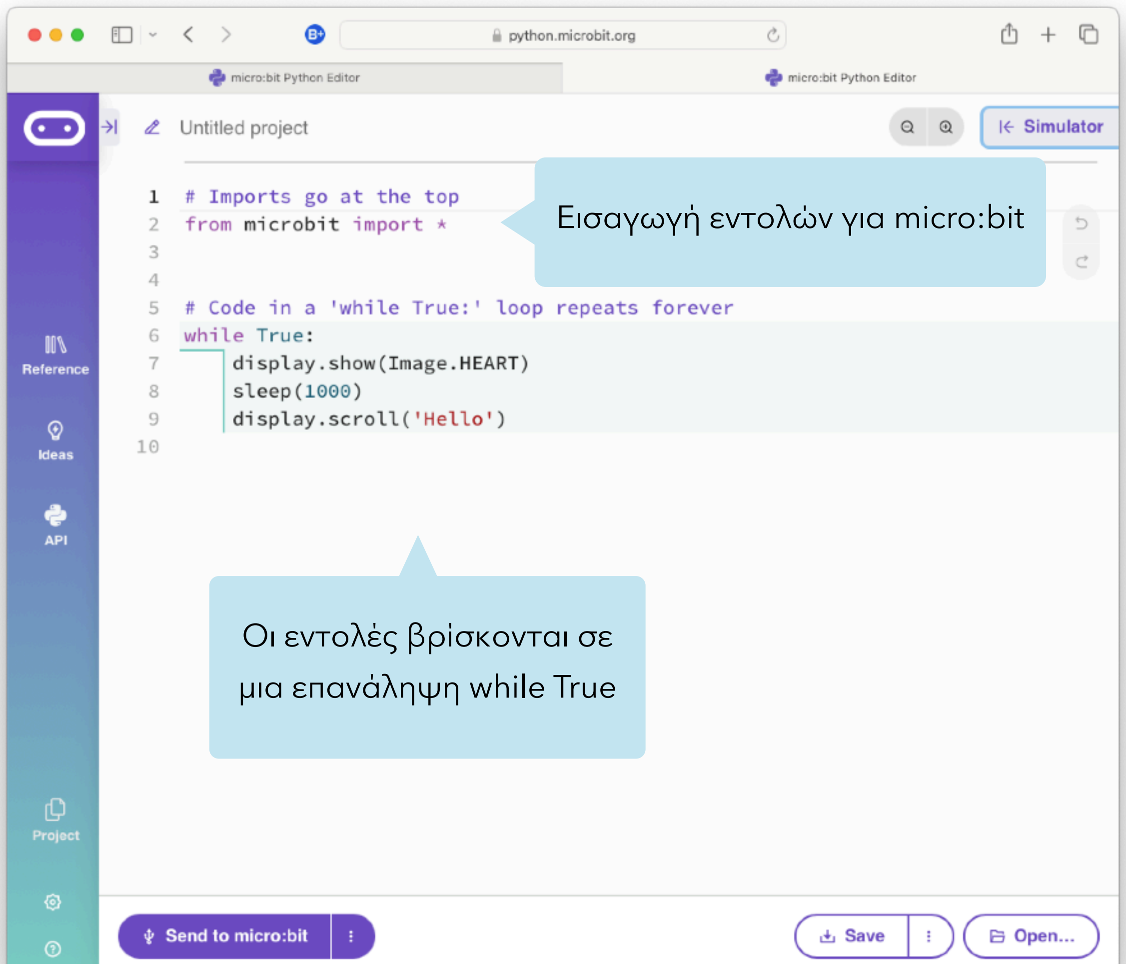
Με τη μεταφορά του στο micro:bit, ξεκινά και η εκτέλεση του. Κάθε φορά που αλλάζουμε τον κώδικα θα πρέπει να στέλνουμε ξανά το πρόγραμμα στο micro:bit!

Εναλλακτικά, μπορούμε από τα Windows (ή το MacOS) να μεταφέρουμε με drag and drop το αρχείο του προγράμματός μας (έχει κατάληξη .hex) στο micro:bit.



Ας γράψουμε κώδικα!

Οι πύθωνες αγαπούν πολύ τις συσκευές. Ειδικά όταν μπορούν να χρησιμοποιούν κώδικα για να τις προγραμματίσουν. Ακόμη περισσότερο, όταν το πρόγραμμα ονομάζεται και Python Editor!



Με την έναρξη νέου έργου, εμφανίζονται εντολές που προβάλλουν πληροφορίες στα 25 LED του micro:bit. Στη συνέχεια θα δούμε τις εντολές αυτές.

```
while True:
```

Η συγκεκριμένη επανάληψη πάντα θα ισχύει, έτσι οι εντολές που περιέχει θα επαναλαμβάνονται συνέχεια (ατέρμονη επανάληψη - infinite loop).

```
display.show(Image.HEART)
```

Το micro:bit υποστηρίζει (και αναγνωρίζει) αριθμό έτοιμων σχημάτων. Αυτή είναι μια πολύ βολική δυνατότητα, καθώς πολύ εύκολα και γρήγορα εμφανίζουμε συγκεκριμένα σχήματα. Εδώ εμφανίζεται μια καρδία.

```
sleep(1000)
```

Με την πιο πάνω εντολή, περιμένει 1000 χιλιοστά του δευτερολέπτου (ή ένα δευτερόλεπτο) πριν εκτελέσει την επόμενη εντολή.

```
display.scroll('Hello')
```

Θα έχετε προσέξει διαφορές ανάμεσα στο Python Editor του micro:bit και στο IDLE που χρησιμοποιήσαμε σε προηγούμενα κεφάλαια. Στο κεφάλαιο αυτό θα χρησιμοποιήσουμε τα χρώματα των εντολών του Python Editor και όχι του IDLE.

Δείξε το με LED!

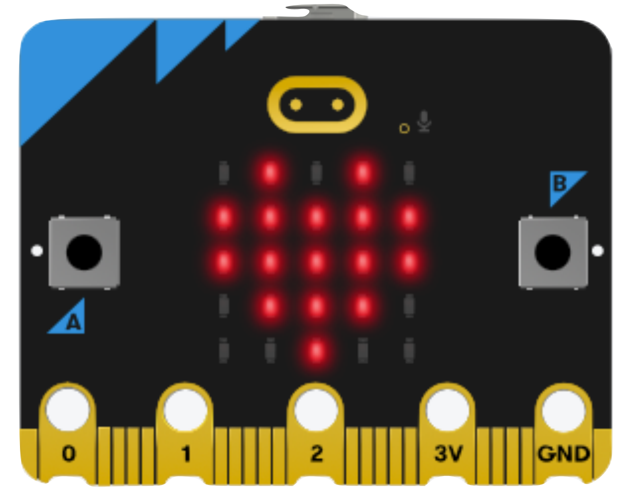
Οι πύθωνες είναι πολύ γλυκά ζώα! Τους αρέσει να αγκαλιάζουν τους πάντες και τα πάντα, μερικές φορές όμως το παρακάνουν! Και εκφράζουν την αγάπη τους με καρδούλες, ειδικά όταν βρουν μπροστά τους LED!

(Σημείωση: δεν πάμε ποτέ να δοκιμάσουμε την αγκαλιά ενός πύθωνα!).

Η εντολή display μπορεί να χρησιμοποιηθεί για να δείξει μια στατική εικόνα ή για να δείξει αριθμούς ή/και κείμενο. Οι πληροφορίες αυτές προβάλλονται με τα LED του micro:bit.



Μια στατική εικόνα (όπως, για παράδειγμα, η καρδούλα) μπορεί εύκολα να προβληθεί με τη χρήση ορισμένων από τα 25 LED του micro:bit



Τα 25 LED (5 σειρές και 5 στήλες) δεν είναι αρκετά για να προβάλουμε κείμενο ή και αριθμούς. Έτσι, για την προβολή τους, οι δημιουργοί του micro:bit έκαναν κάτι πολύ έξυπνο: το κείμενο (ή/και οι αριθμοί) "κυλούν" (κάνουν scroll) στα 25 LED!

Για να προβάλουμε είτε (στατική) εικόνα, είτε κείμενο (ή αριθμούς), ξεκινάμε πάντα με την εντολή display. Στη συνέχεια, αν θα προβληθεί στατική εικόνα, πληκτρολογούμε το show και σε παρένθεση μια από τις έτοιμες εικόνες (σημείωση: σε επόμενη σελίδα θα δούμε πώς δημιουργούμε τις δικές μας). Η εντολή είναι:

```
display.show(Image.HEART).
```

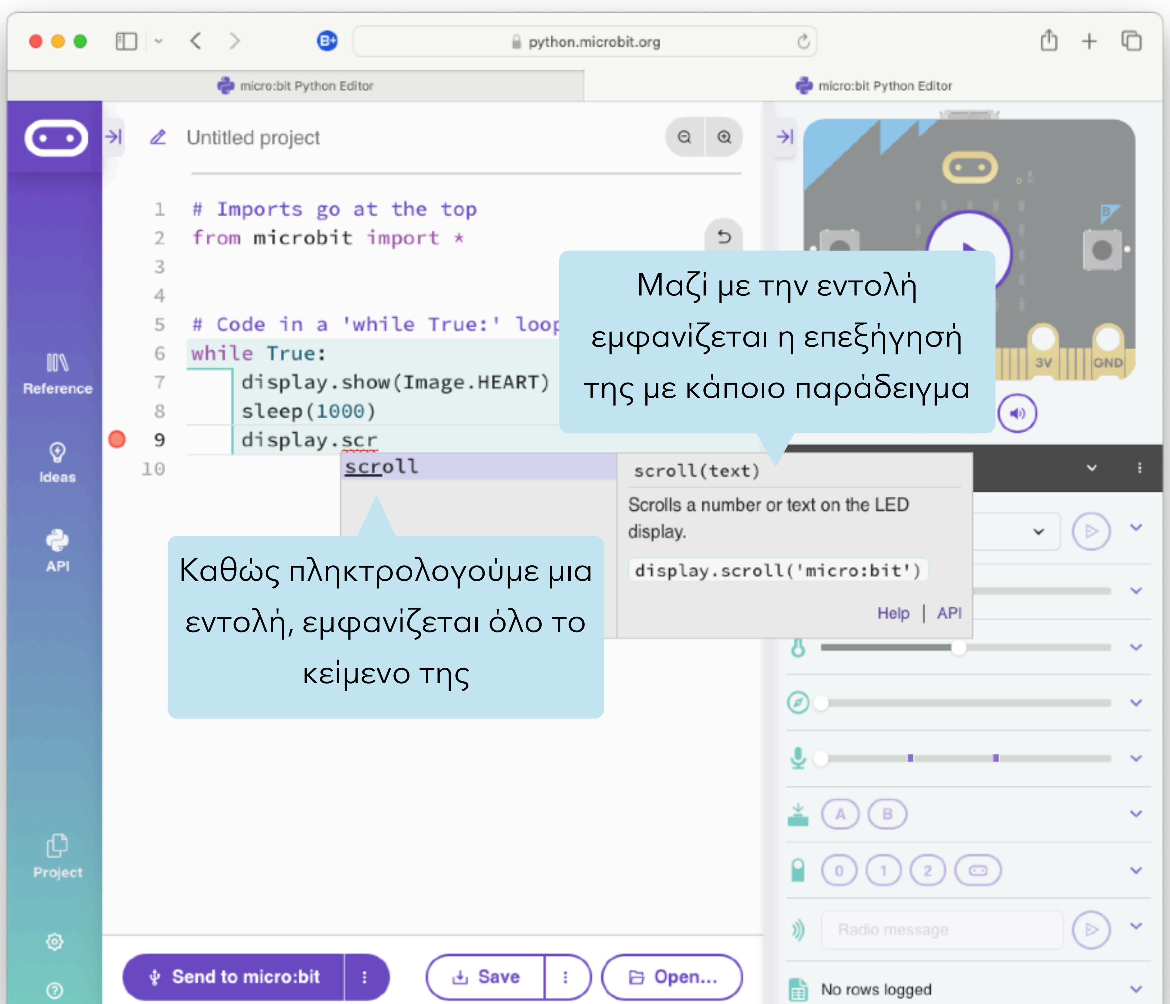
Για κείμενο ή αριθμό, ξεκινάμε με την display, αλλά στη συνέχεια προσθέτουμε το scroll και σε παρένθεση το κείμενο. Η εντολή είναι:

```
display.scroll('Hello')
```

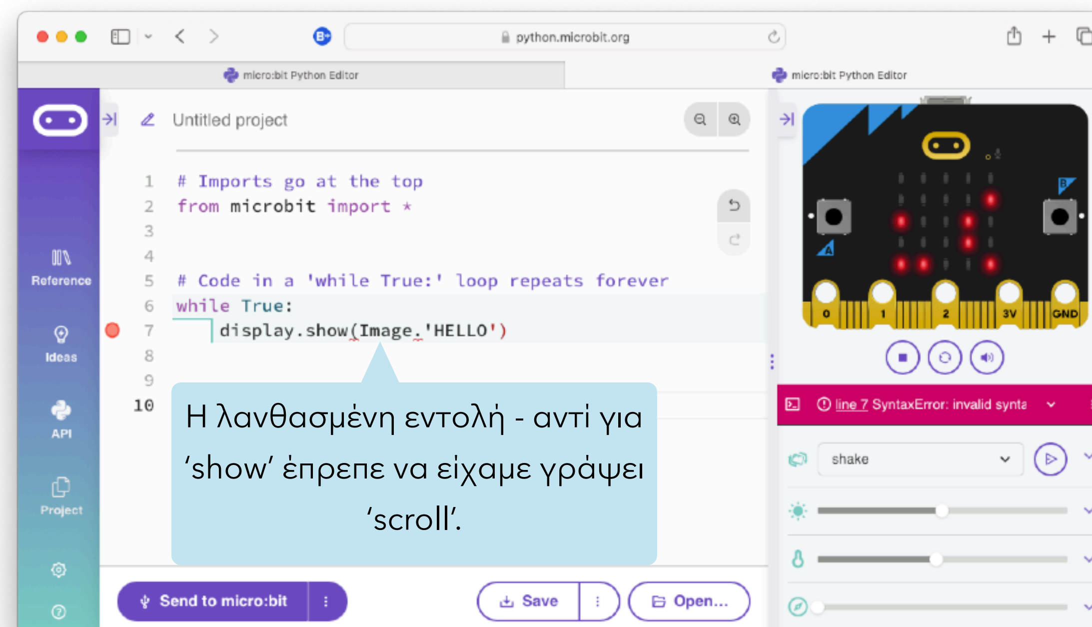
Τα λάθη είναι πυθώνια…

(Οκ δεν υπάρχει τέτοια λέξη -πυθώνια- αλλά… ας μιλήσουμε για λάθη στον κώδικα Python για micro:bit).

Το Python Editor μας βοηθά στην πληκτρολόγηση των εντολών. Καθώς πληκτρολογούμε μια εντολή, εμφανίζεται επιλογή με τον τρόπο γραφής της (ώστε να αποφύγουμε λάθη στην πληκτρολόγηση). Αν η εντολή αποτελείται και από άλλα μέρη (π.χ. display), τότε εμφανίζονται και οι επιλογές της (εικόνα κάτω).



Μια εντολή, όμως, δεν αρκεί να είναι σωστά γραμμένη (π.χ. "ορθογραφικά"). Θα πρέπει και η σύνταξη της να είναι σωστή. Για παράδειγμα, όπως είδαμε σε προηγούμενη σελίδα, όταν θέλουμε να εμφανίσουμε κείμενο, χρησιμοποιούμε την display.scroll. Το κείμενο θα "κυλήσει" στα LED. Κείμενο μπορούμε να προβάλουμε και με την display.show, όμως θα εμφανίζεται ένας χαρακτήρας κάθε φορά. Αν χρησιμοποιήσουμε την display.show με την Image στην παρένθεση, τότε έχουμε συντακτικό λάθος και θα εμφανιστεί σχετικό μήνυμα.



Το Python Editor είναι πολύ πιο βοηθητικό στον εντοπισμό λαθών από το IDLE. Και είναι φυσικό, καθώς μας ευκολύνει να μάθουμε προγραμματισμό!

Ας “ρίξουμε” εντολές

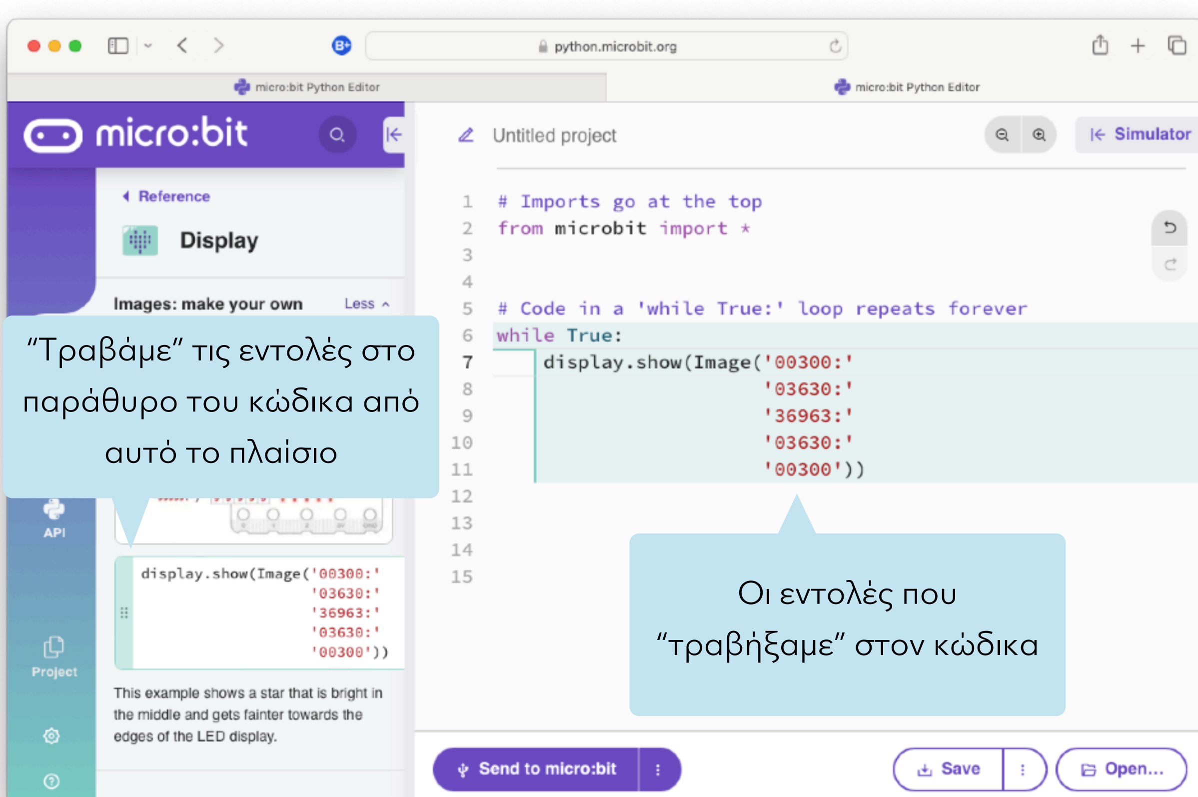
Για να αποφύγουμε συντακτικά ή ορθογραφικά λάθη στον κώδικα, ή απλά αν δυσκολευόμαστε στην πληκτρολόγηση, μπορούμε να χρησιμοποιήσουμε τις έτοιμες εντολές. Κάνουμε κλικ στο Reference για να εμφανιστούν οι κατηγορίες. Οι κατηγορίες αυτές αφορούν κυρίως τις λειτουργίες κάθε μέρους του micro:bit. Από την Display, που αφορά τα LED, στα κουμπιά, στους αισθητήρες, αλλά και σε πιο πολύπλοκα θέματα όπως τη λήψη αποφάσεων, τις επαναλήψεις κτλ.

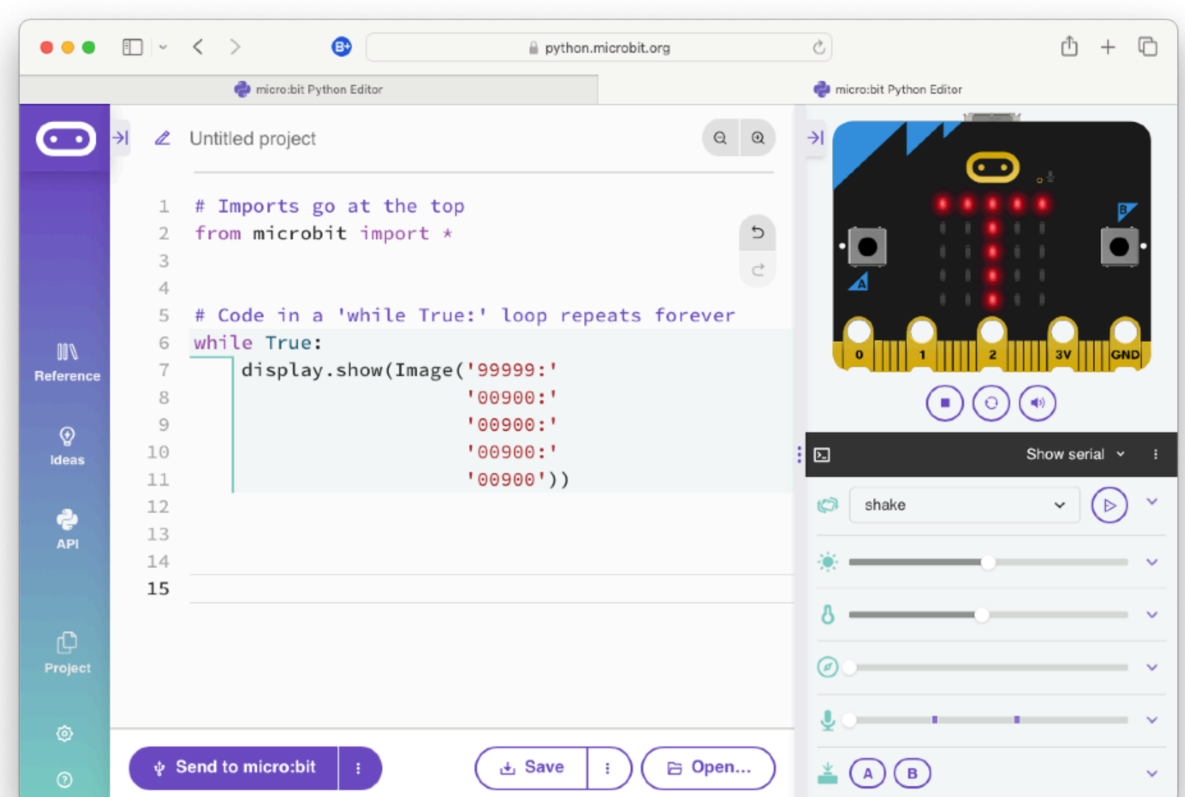
Επιλέγουμε την κατηγορία **Display** και στη συνέχεια την **Images: make your own** και την “τραβάμε” στον κώδικα.

Όπως φαίνεται στην εικόνα (κάτω αριστερά), εμφανίζονται 5 σειρές με αριθμούς σε 5 στήλες. Κάθε αριθμός αντιπροσωπεύει και ένα LED του micro:bit. Επίσης, κάθε αριθμός δείχνει και το πόσο φωτεινό θα είναι το LED. Αν ο αριθμός είναι 0, σημαίνει πως το LED δε θα λειτουργεί. Αν είναι στο 9, σημαίνει πως θα είναι στο μέγιστο της φωτεινότητας του.

Θα σχηματίσουμε ένα φωτεινό “Τ” με τα LED:

Στην πρώτη γραμμή των LED βάζουμε μόνο το ‘9’. Στις υπόλοιπες γραμμές θέτουμε όλα τα LED στο 0 (σβηστά) και τη μεσαία στήλη στο 9. Εμφανίζεται το γράμμα “Τ” (εικόνα κάτω).

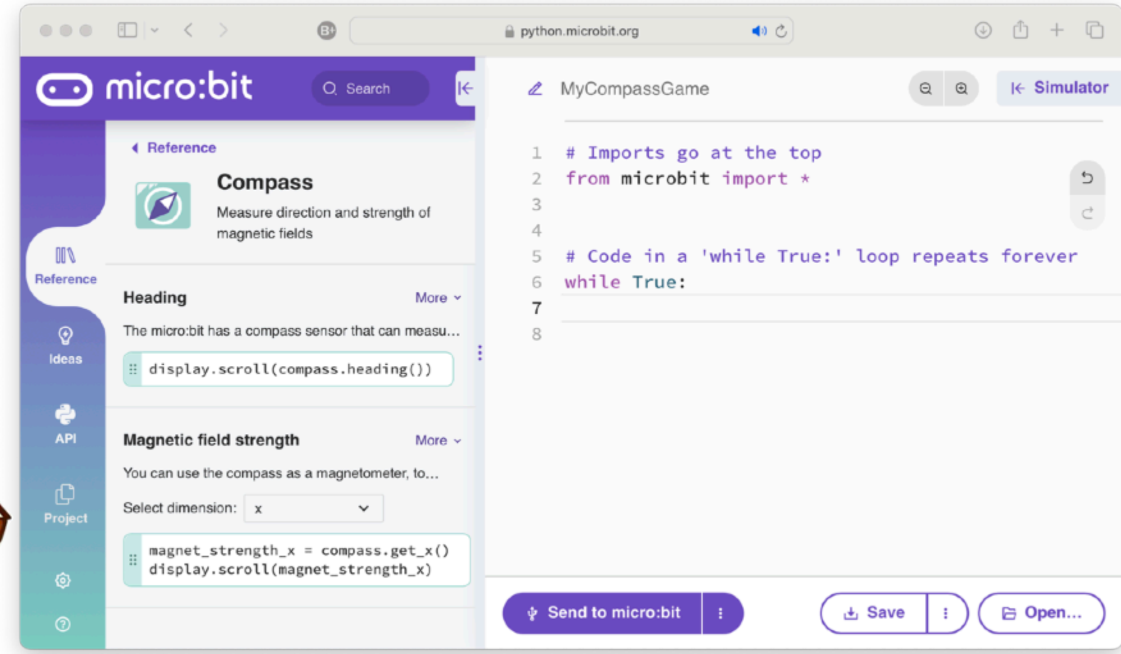




Πυξίδες & πύθωνες!

Ο φίλος μας ο πύθωνας αποφάσισε να παίξει ένα παιχνίδι: έκρυψε ένα μικρό θησαυρό σε κάποιο σημείο της αυλής του σχολείου (μπορείτε να το παίξετε και εσείς αυτό). Μας ζήτησε να ξεκινήσουμε από ένα σημείο και να ακολουθήσουμε τις οδηγίες του για να βρούμε τον θησαυρό.





Οι οδηγίες λένε να προχωρήσουμε κάποια βήματα (βάλτε εσείς τα πόσα βήματα θέλετε) Βόρεια, μετά να πάμε Ανατολικά κάποια βήματα, στη συνέχεια να πάμε ξανά Βόρεια, και τέλος να προχωρήσουμε Δυτικά.

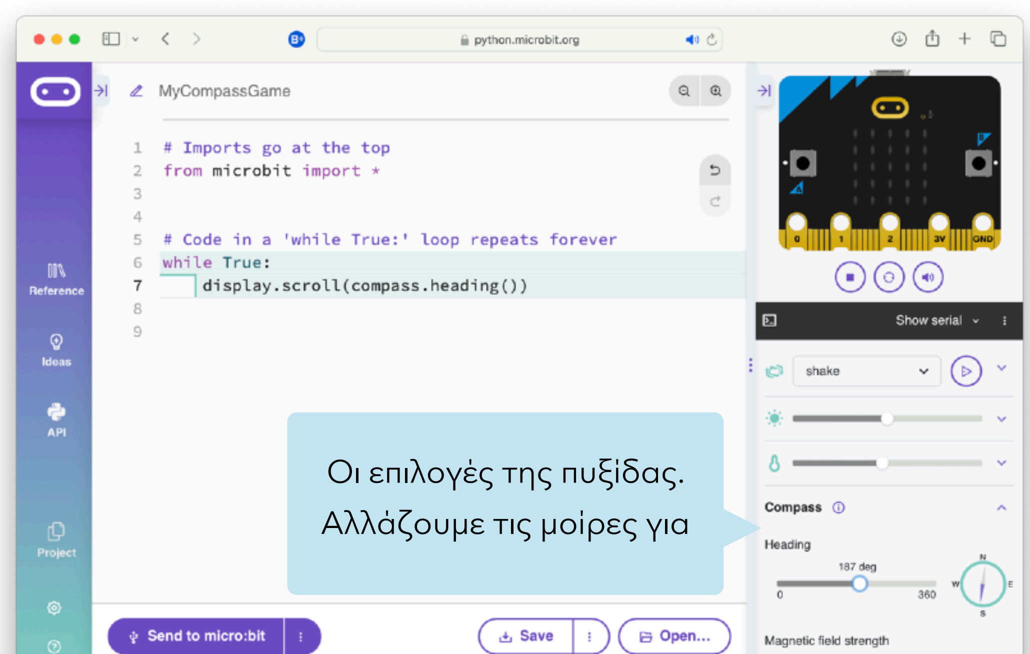
Μπερδευτικό κάπως, αλλά ευτυχώς που τα micro:bit έχουν και πυξίδα!



Παίξτε αυτό το παιχνίδι στο σχολείο - δώστε στους φίλους σας οδηγίες με βήματα και κατευθύνσεις Βόρεια - Νότια - Ανατολικά - Δυτικά.

Για να δούμε τις εντολές της πυξίδας μπορούμε να ανοίξουμε το Reference και να επιλέξουμε Compass.

Η εντολή display.scroll(compass.heading()) προβάλλει στο micro:bit μας με αριθμό την κατεύθυνση στην οποία "βλέπει" το micro:bit μας. Ο αριθμός ξεκινά από το 0 (μοίρες) για τον Βορρά (North), 90 για Ανατολή, 180 για Νότο, 270 για Δύση και 360 (ξανά) για Βορρά.



Η εμφάνιση σε μοίρες είναι αρκετά χρήσιμη, όμως δεν είναι και τόσο κατανοητή. Είναι πιο απλό να βλέπουμε το βελάκι που δείχνει τον βορρά, για παράδειγμα, όπως κάνουν οι περισσότερες πυξίδες.

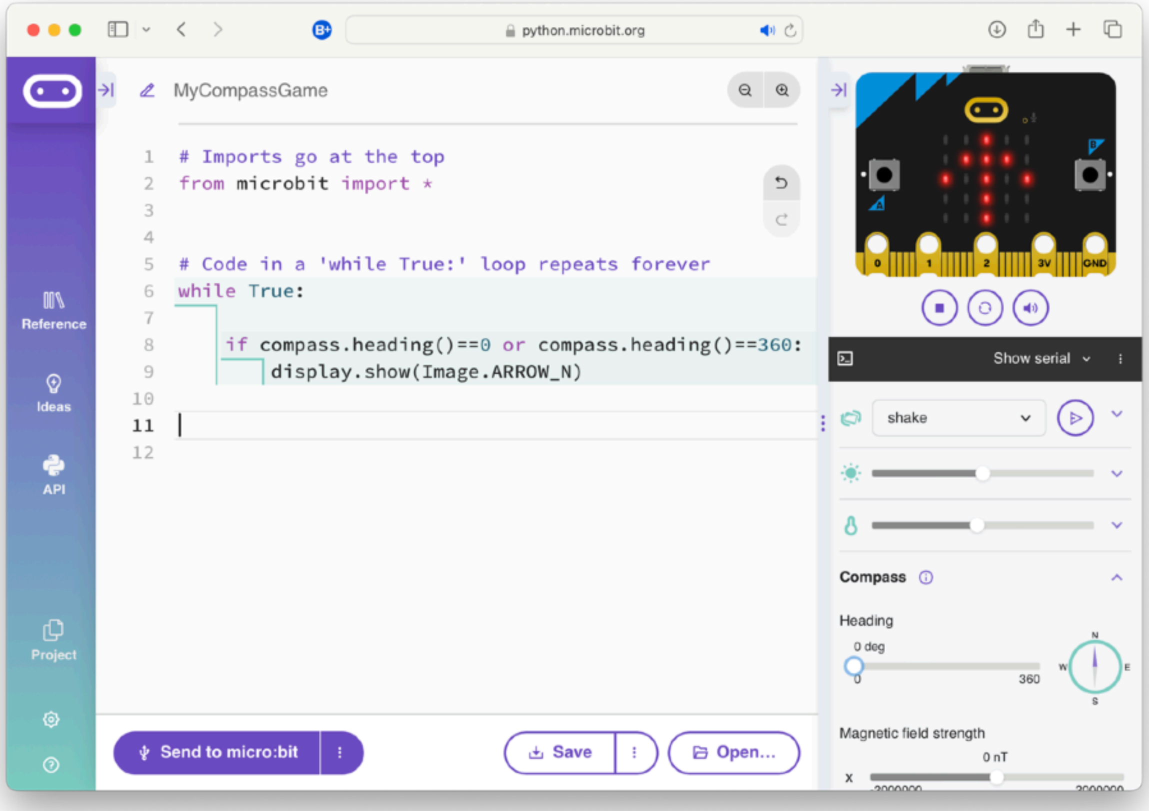
Η εντολή που δώσαμε (προηγούμενη σελίδα) εμφανίζει στο micro:bit την ένδειξη με αριθμούς (σε μοίρες). Εμείς θέλουμε να εμφανίζει τον Βορρά αν είμαστε στις 0° ή 360°.

```
if compass.heading()==0 or compass.heading()==360:
    display.show(IMAGE.ARROW_N)
```

Με την εντολή if ελέγχουμε κατά πόσο η τιμή του compass είναι το 0 ή το 360. Και οι δύο τιμές αντιστοιχούν στη θέση του Βορρά. Σε περίπτωση που ισχύει η συνθήκη, εκτελείται η εντολή που προβάλλει το βέλος στον Βορρά (εικόνα πάνω δεξιά).

Με παρόμοιο τρόπο μπορούμε να προσθέσουμε εντολές με if ώστε να προβάλουμε βέλος για Νότο, Ανατολή κ.α.

Αν γνωρίζουμε ένα σημείο του ορίζοντα (π.χ. Βορρά), τότε πολύ εύκολα μπορούμε να βρούμε τα άλλα (Νότο, Δύση, Ανατολή).



Στην εικόνα πάνω μπορούμε να ελέγξουμε τη λειτουργία του κώδικα μας. Μετακινούμε τις μοίρες της πυξίδας ώστε να δούμε την αλλαγή στο βέλος που προβάλλεται στο micro:bit.

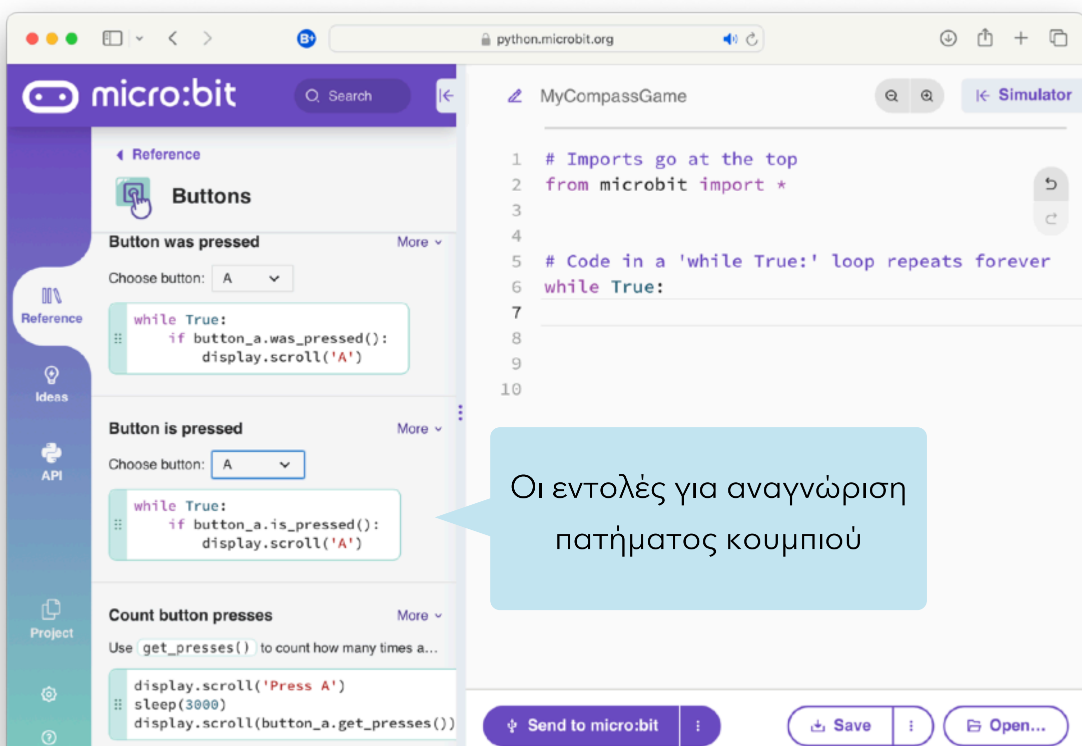


Ας πατήσουμε κουμπιά...

Οι πύθωνες έχουν μια ιδιαίτερη αγάπη για τα κουμπιά. Τους αρέσει να τα πατούν, όσο μικρά και αν είναι, ιδιαίτερα αν είναι στρογγυλά.

Το micro:bit έχει στο μπροστινό του μέρος 2 κουμπιά (το “A” και το “B”). Μπορούμε να τα προγραμματίσουμε ώστε να εκτελούν εντολές όταν πατάμε το “A” ή το “B” ή και τα δύο μαζί!

Το πρώτο που πρέπει να κάνουμε, είναι να γράψουμε τις εντολές για να καταλαβαίνει το micro:bit ποιο κουμπί πατήσαμε και τι θέλουμε να κάνει!



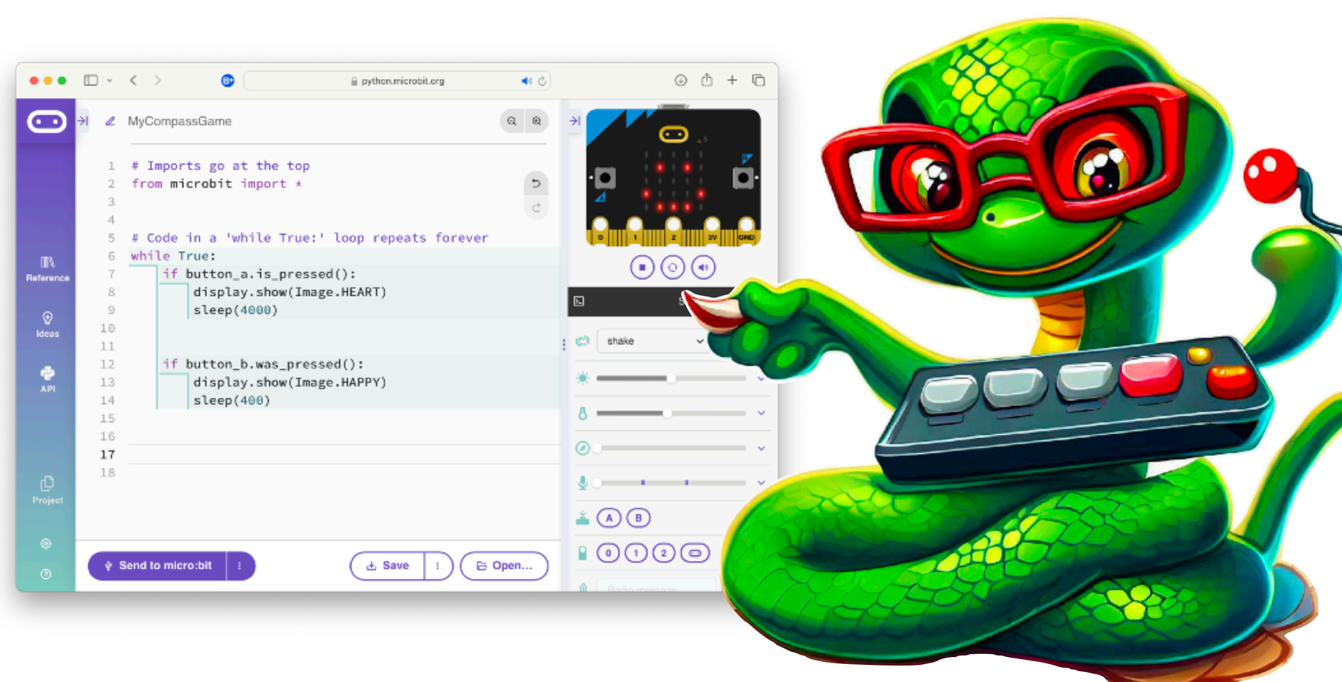
Με τη πιο κάτω συνθήκη, ελέγχουμε αν πατήθηκε κάποιο κουμπί (στο παράδειγμα εδώ το “A”). Αν έχει πατηθεί, τότε εκτελούνται οι εντολές της συνθήκης:

```
If button_a.is_pressed():
    display.show(Image.HEART)
    sleep(4000)
```

Με το πάτημα του κουμπιού “A”, θα εμφανιστεί η εικόνα της καρδιάς στο micro:bit. Θα προσθέσουμε και εντολές για το κουμπί “B”.

```
If button_b.is_pressed():
    display.show(Image.HAPPY)
    sleep(4000)
```

Με την εκτέλεση του κώδικα, είτε σε πραγματικό micro:bit, είτε στο εικονικό, θα δούμε πως με το πάτημα του “A” εμφανίζεται μια καρδία, ενώ με το πάτημα του “B” εμφανίζεται χαρούμενο πρόσωπο.



Πέτρα, Ψαλίδι, Χαρτί

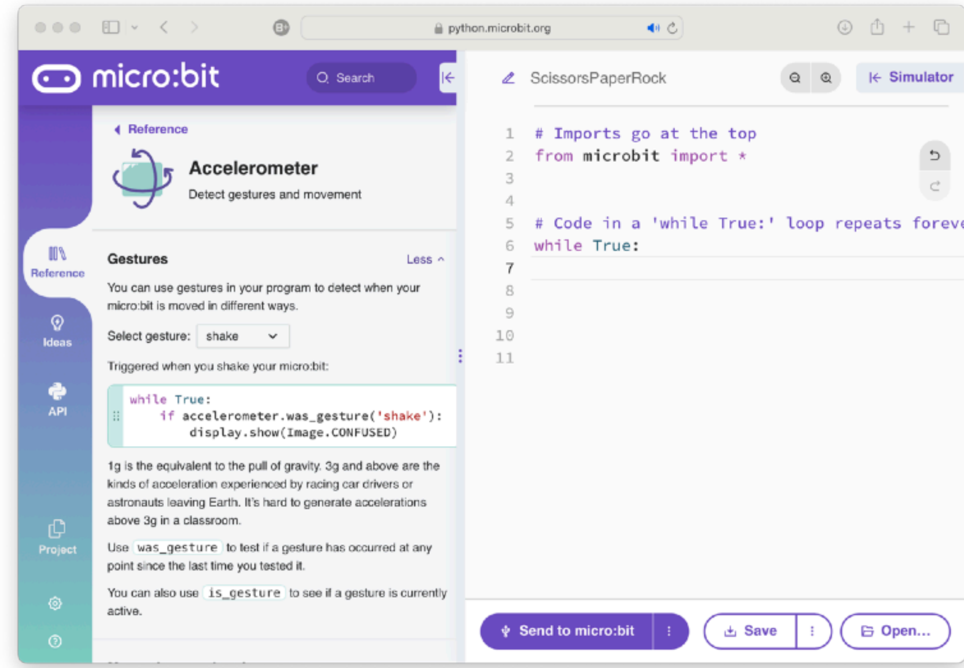
Γνωρίζουμε πόσο παιχνιδιάρικα ζωάκια είναι οι πύθωνες. Όταν βρεθούν δύο μαζί, πάντα παίζουν το αγαπημένο τους παιχνίδι: "Πέτρα, Ψαλίδι, Χαρτί".

Με τη βοήθεια του micro:bit θα δημιουργήσουμε ένα τέτοιο παιχνίδι. Θα χρησιμοποιήσουμε το επιταχυνσιόμετρο (μεγάλη λέξη, σωστά;) του micro:bit. Είναι ένας αισθητήρας που βοηθά το micro:bit μας να "καταλάβει" αν πέφτει, προς ποια μεριά το γυρνάμε, αν το κουνάμε κτλ.

Εμείς θα χρησιμοποιήσουμε τον αισθητήρα (accelerometer στην αγγλική) για να εμφανίζει το micro:bit εικόνες στα LED όταν το κουνάμε (shake).

Στην εικόνα πάνω δεξιά εμφανίζονται οι επιλογές για το επιταχυνσιόμετρο (accelerometer).





Η συνθήκη για λειτουργία του αισθητήρα είναι:

```
if accelerometer.was_gesture('shake'):
    display.show(Image.CONFUSED)
```

Με την πιο πάνω συνθήκη, όταν κουνήσουμε το micro:bit, θα εμφανιστεί στα LED η εικόνα που επιλέξαμε (confused).



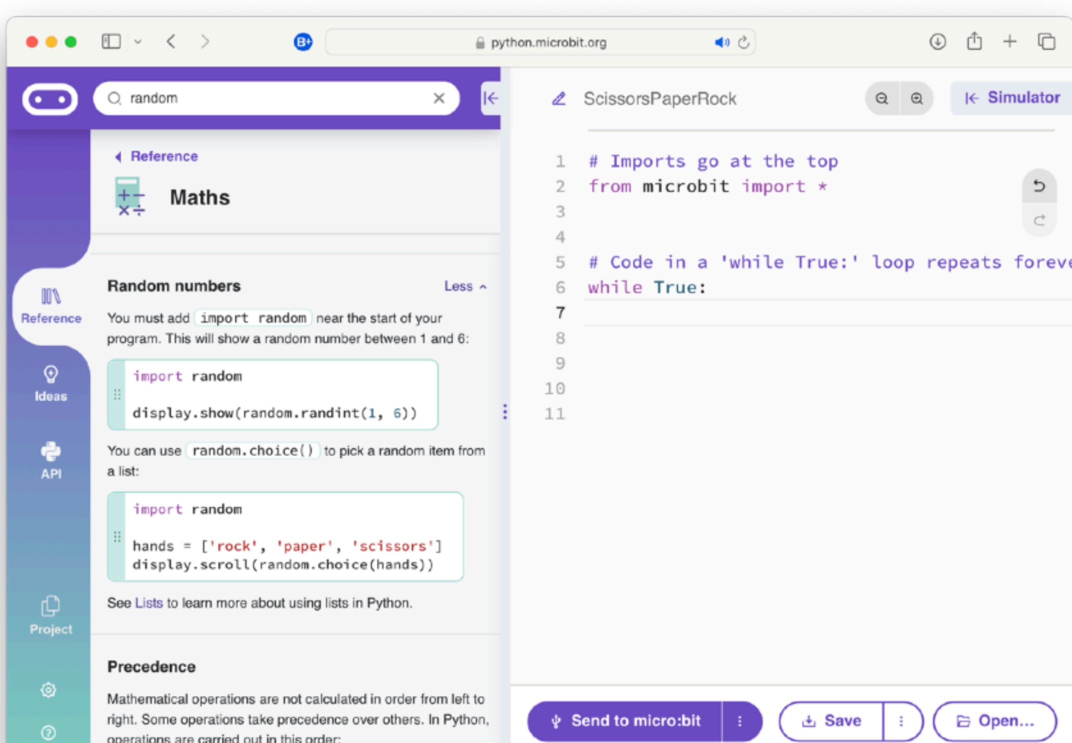
Το επιταχυνσιόμετρο μπορεί να "καταλάβει" πότε γυρνάμε το micro:bit, πότε το αφήνουμε να πέσει (μην το δοκιμάσετε στο έδαφος!), προς ποια μεριά το περιστρέφουμε κ.α. Αυτό επιτρέπει τη δημιουργία πολύπλοκου κώδικα όπου, ανάλογα με τη θέση του micro:bit, θα εκτελούμε και διαφορετικές εντολές.

Τυχαίες επιλογές…

Με τον κώδικα της προηγούμενης σελίδας, όταν κουνάμε (shake) το micro:bit, εμφανίζεται συγκεκριμένη εικόνα στα LED. Όμως, αυτό δεν είναι αρκετό. Στο παιχνίδι, μετράμε μέχρι το 3 (ή λέμε “πέτρα, ψαλίδι, χαρτί”) και ανάλογα με το τι θα φέρουμε, κερδίζουμε ή χάνουμε.

Αυτό που πρέπει να κάνουμε είναι να εμφανίζουμε μια από τις 3 εικόνες τυχαία! Για να δούμε πώς γίνεται αυτό με την Python για micro:bit…

Για να μπορέσουμε να κάνουμε τυχαίες επιλογές, θα πρέπει να κάνουμε κάποιες αλλαγές στον κώδικα.



Για να μπορούμε να χρησιμοποιήσουμε τυχαίους αριθμούς, θα πρέπει να ξεκινήσουμε με την εντολή “import random”.

Τώρα χρειαζόμαστε 2 πράγματα: μια μεταβλητή, που να παίρνει τιμές από το 1 ως το 3 (ένας αριθμός για κάθε αντικείμενο - πέτρα, ψαλίδι, χαρτί).

```
number=random.randint(1,3)
```

Με την πιο πάνω εντολή, η μεταβλητή number παίρνει μια τιμή από το 1 μέχρι το 3. Τώρα θα πρέπει να εμφανίζουμε εικόνα για κάθε αριθμό (π.χ. ψαλίδι αν έρθει ο αριθμός 1). Η εικόνα θα πρέπει να εμφανίζεται όταν κουνάμε το micro:bit.

```
while True: #ατέρμονη επανάληψη
    if accelerometer.was_gesture(‘shake’)
```

Ελέγχουμε αν έχουμε κουνήσει το micro:bit με την πιο πάνω εντολή. Στη συνέχεια θα δώσουμε εντολή ώστε να επιλέγεται ένας τυχαίος αριθμός από το 1 ως το 3:

```
number=random.randint(1,3)
```

Η μεταβλητή number θα πάρει μια τιμή από το 1 ως το 3.

Μέχρι αυτό το σημείο, ελέγχουμε αν έχει ενεργοποιηθεί ο αισθητήρας με το κούνημα ('shake'). Αν έχουμε κουνήσει το micro:bit, τότε η μεταβλητή number παίρνει έναν τυχαίο αριθμό από το 1-3.

Στη συνέχεια θα δούμε τι συμβαίνει σε κάθε περίπτωση.

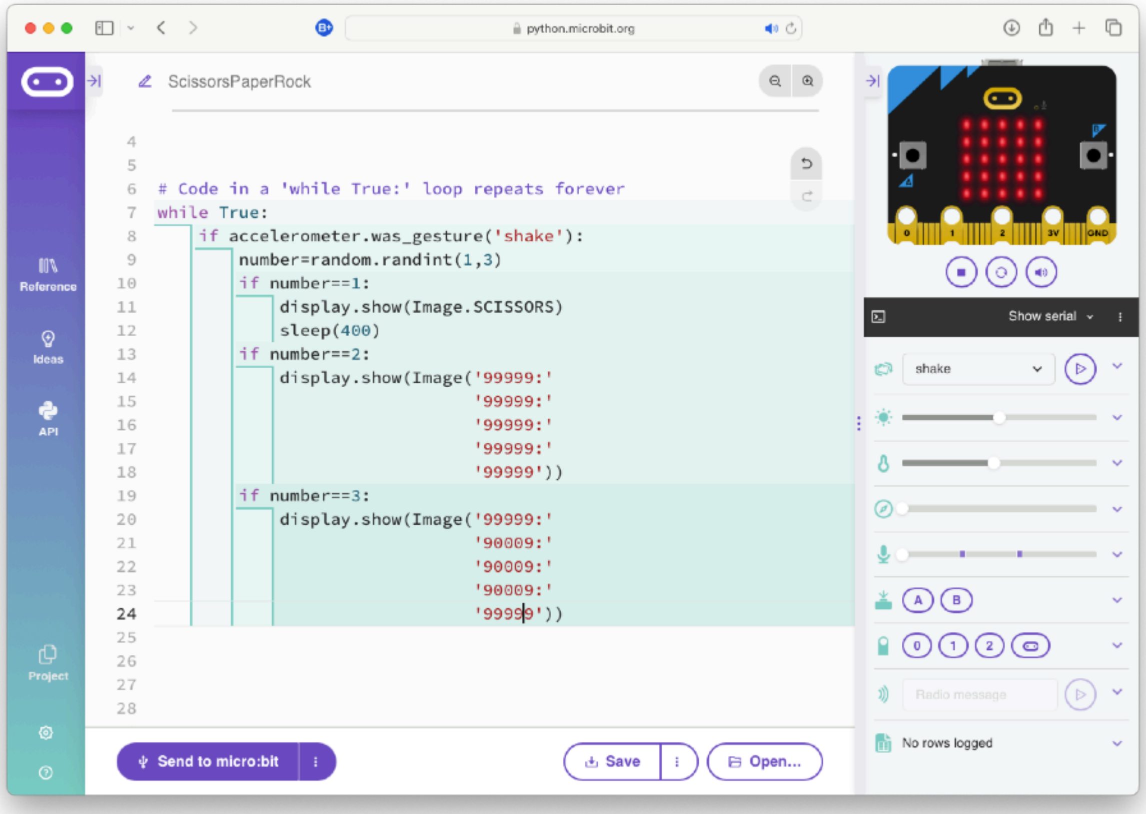
```
If number=1:
    display.show(Image.SCISSORS)
    sleep(1000)
```

Αν ο τυχαίος αριθμός είναι το 1, τότε θα εμφανιστεί η εικόνα του ψαλιδιού. Θα συνεχίσουμε με ακόμη 2 συνθήκες.

```
if number=2:
    display.show(Image.('99999'
                        '99999'
                        '99999'
                        '99999'
                        '99999'))
    sleep(1000)
```

Επειδή δεν υπάρχει έτοιμη εικόνα για την πέτρα, δημιουργούμε το δικό μας σχήμα (όλα τα LED ανοικτά).

Μπορούμε να ελέγξουμε τον κώδικα από το εικονικό micro:bit. Κάνουμε κλικ στο αντίστοιχο εργαλείο.



Στην εικόνα πιο πάνω βλέπουμε όλο τον κώδικα που γράψαμε. Αν χρησιμοποιήσουμε δύο micro:bits, τότε μπορούμε να παίξουμε εύκολα το "Πέτρα, Ψαλίδι, Χαρτί".

Με τη random μπορούμε να δημιουργήσουμε πολλές δραστηριότητες και παιχνίδια. Μπορούμε να δημιουργήσουμε ακόμη και ζάρι.



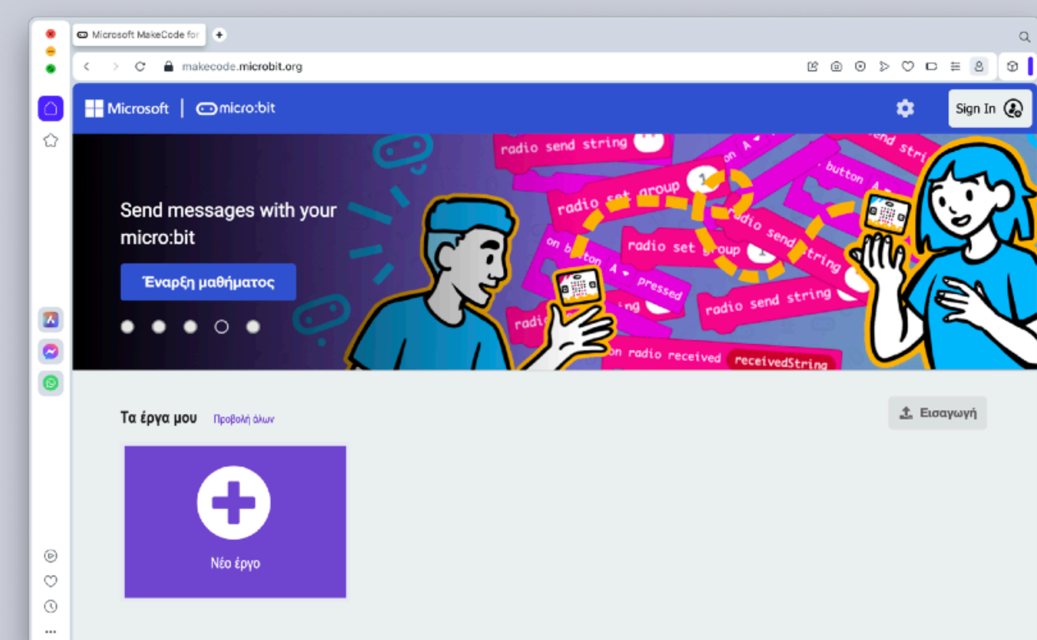
```
number=random.randint(1,6)
display.show(number)
```

Το περιβάλλον MakeCode

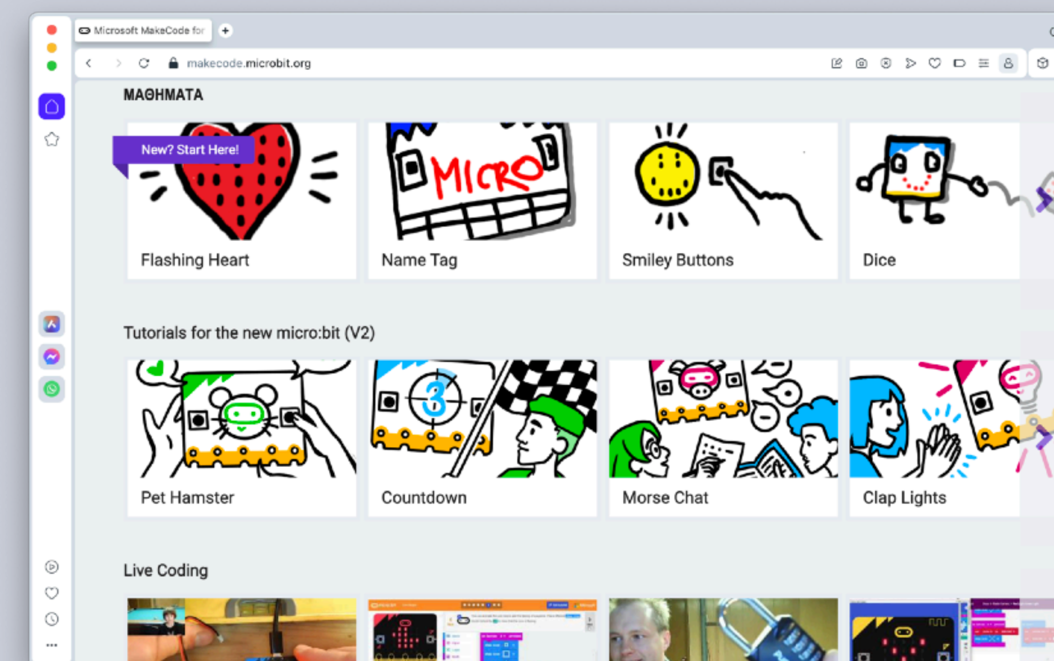
Το MakeCode εναλλακτικό, και ιδιαίτερα φιλικό περιβάλλον προγραματισμού του micro:bit του. Προσφέρει αρκετές ευκολίες, συμπεριλαμβανομένης της επιλογής να προγραμματιστεί αποκλειστικά με μπλοκ.

Περιλαμβάνει μεγάλη σειρά έτοιμων μαθημάτων που μπορούν να μας βοηθήσουν να γνωρίσουμε τις βασικές λειτουργίες του micro:bit, και καθώς βελτιώνουμε τις γνώσεις μας, να δημιουργήσουμε και πιο πολύπλοκα προγράμματα.

Το περιβάλλον προγραμματισμού μπορούμε να το κατεβάσουμε στον υπολογιστή μας και να το εγκαταστήσουμε, ή να εργαστούμε αποκλειστικά μέσω διαδικτύου, από τη διεύθυνση: https://makecode.microbit.org



Υποστηρίζονται πολλές γλώσσες, ανάμεσά τους και η ελληνική, κάτι που ευκολύνει τη λειτουργία του και με πιο μικρά παιδιά.

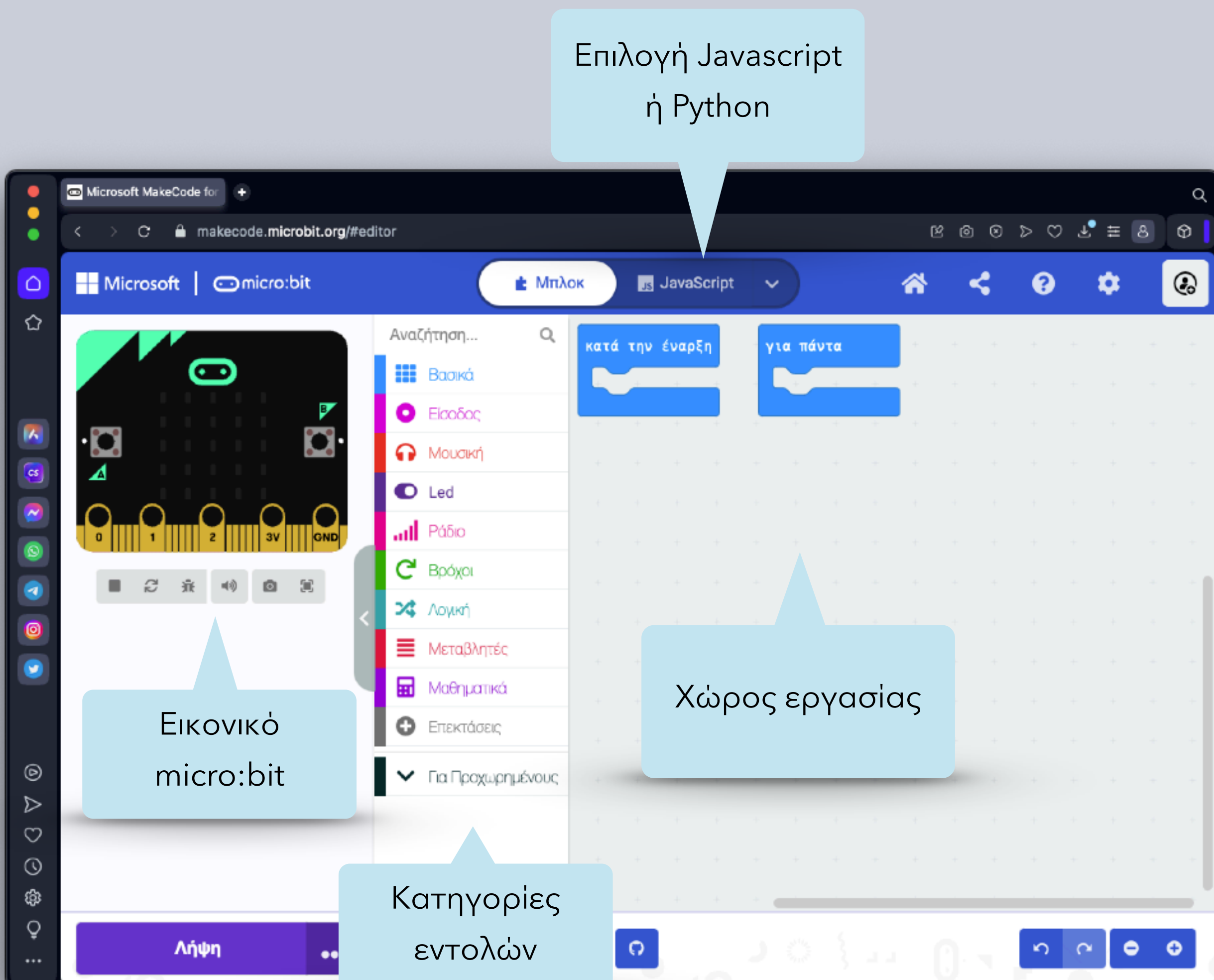


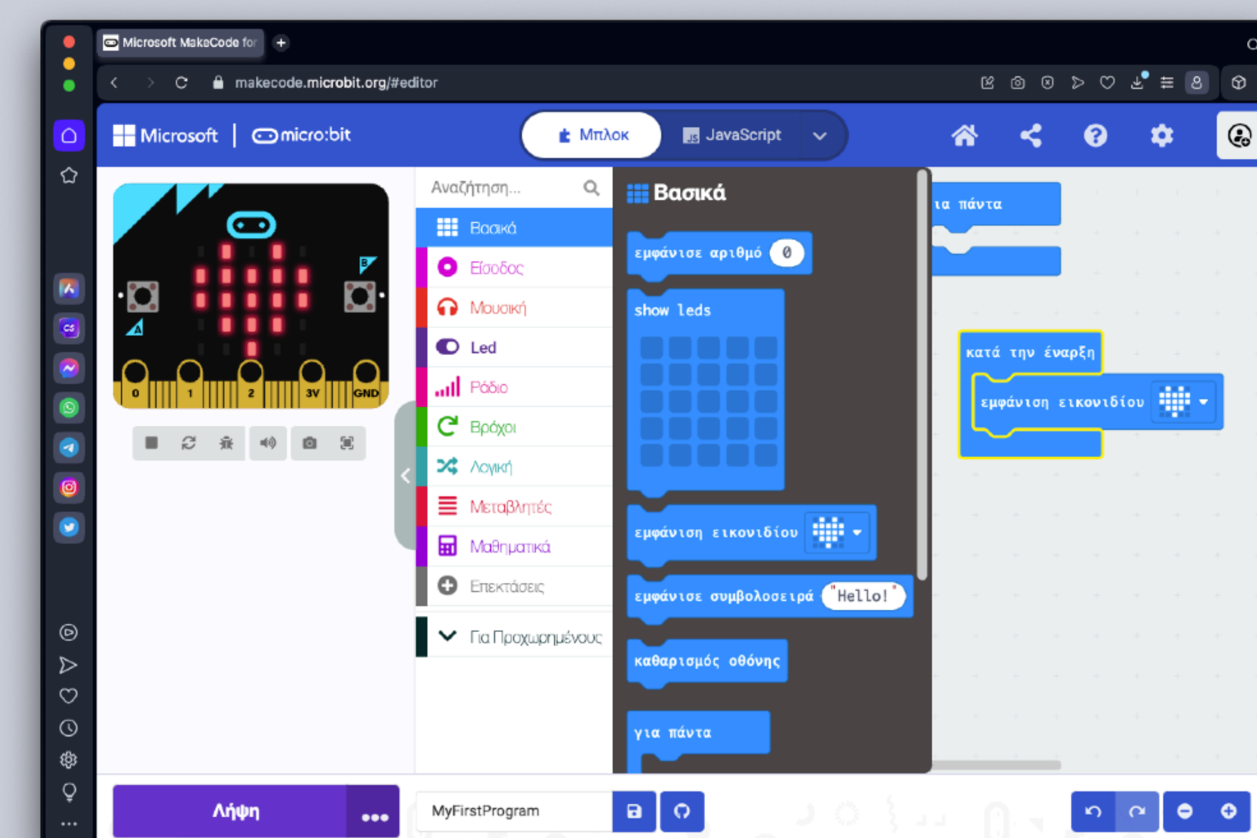
“Εικονικό” micro:bit

Το MakeCode εμφανίζει ένα εικονικό micro:bit (εικόνα κάτω) το οποίο συμπεριφέρεται όπως και το πραγματικό. Μπορούμε να πατήσουμε τα κουμπιά του, να το ‘κινήσουμε’ για να δοκιμάσουμε τους αισθητήρες του, και γενικά να εφαρμόσουμε όλες (σχεδόν) τις λειτουργίες του πραγματικού. Αυτό είναι ιδιαίτερα χρήσιμο, καθώς μπορούμε να μάθουμε τη λειτουργία του, και να δημιουργήσουμε σύνθετα προγράμματα με πολλές γραμμές κώδικα, ακόμη και αν δεν έχουμε το micro:bit.

Ο προγραμματισμός με μπλοκ είναι πολύ εύκολος, μιας και όλες οι εντολές εμφανίζονται κάτω από τις κατηγορίες τους. Κάθε κατηγορία επίσης έχει το δικό της χρώμα. Στον χώρο εργασίας (εικόνα κάτω αριστερά), με τη δημιουργία ενός νέου προγράμματος (έργου), εμφανίζονται δύο μπλοκ. Το ένα “με την έναρξη” εκτελεί τις εντολές που θα του δώσουμε, καθώς εκκινεί για πρώτη φορά το micro:bit. Το δεύτερο (“για πάντα”) εκτελεί συνεχώς τις εντολές που περιέχει.

Οι Βασικές εντολές (μπλε χρώμα) σχετίζονται με το περιεχόμενο που θα εμφανίζεται στα 25 LED (χωρισμένα σε 5 σειρές και 5 στήλες) του micro:bit. Ανάλογα με την επιλογή μας, το αποτέλεσμα εμφανίζεται στο micro:bit.





Βασικές εντολές micro:bit

Αν και δεν είναι στόχος του οδηγού αυτού η εκμάθηση του micro:bit με τη χρήση μπλοκ, θα γνωρίσουμε τις βασικότερες λειτουργίες και εντολές.

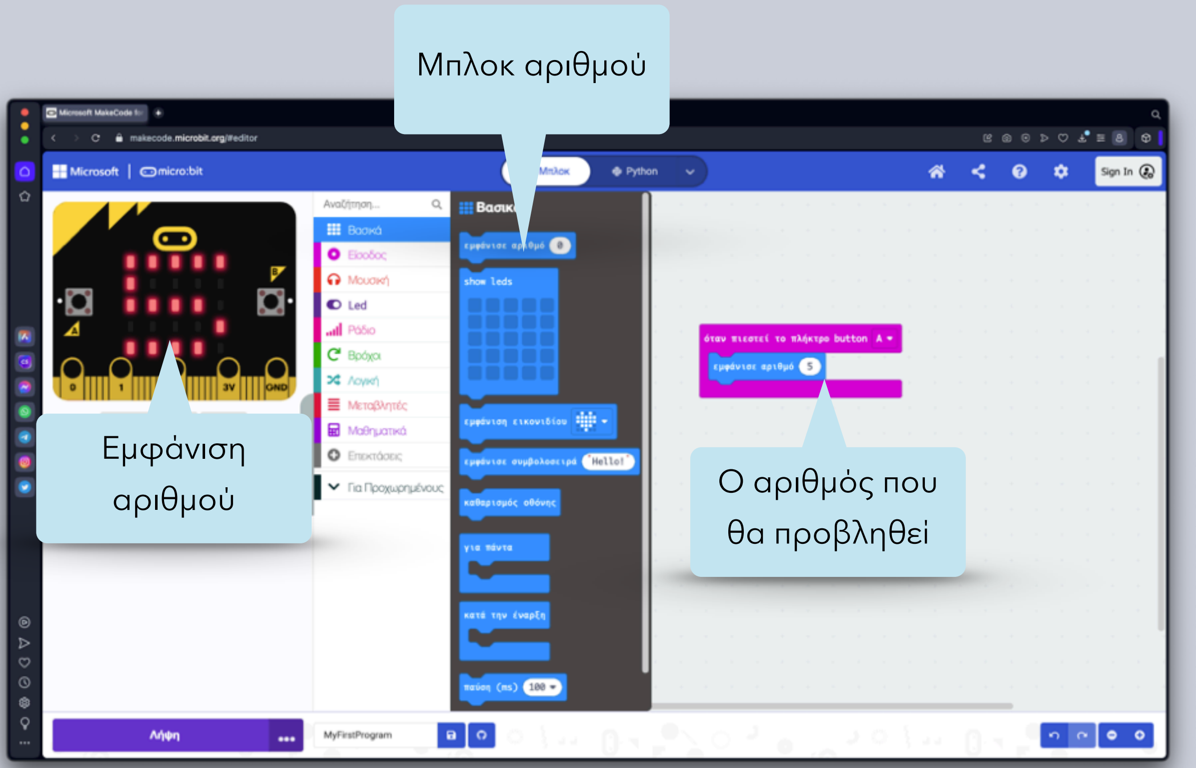
Από την κατηγορία 'Είσοδος", επιλέγουμε "όταν πιεστεί το πλήκτρο Α" και το μεταφέρουμε στον χώρο εργασίας (εικόνα κάτω).

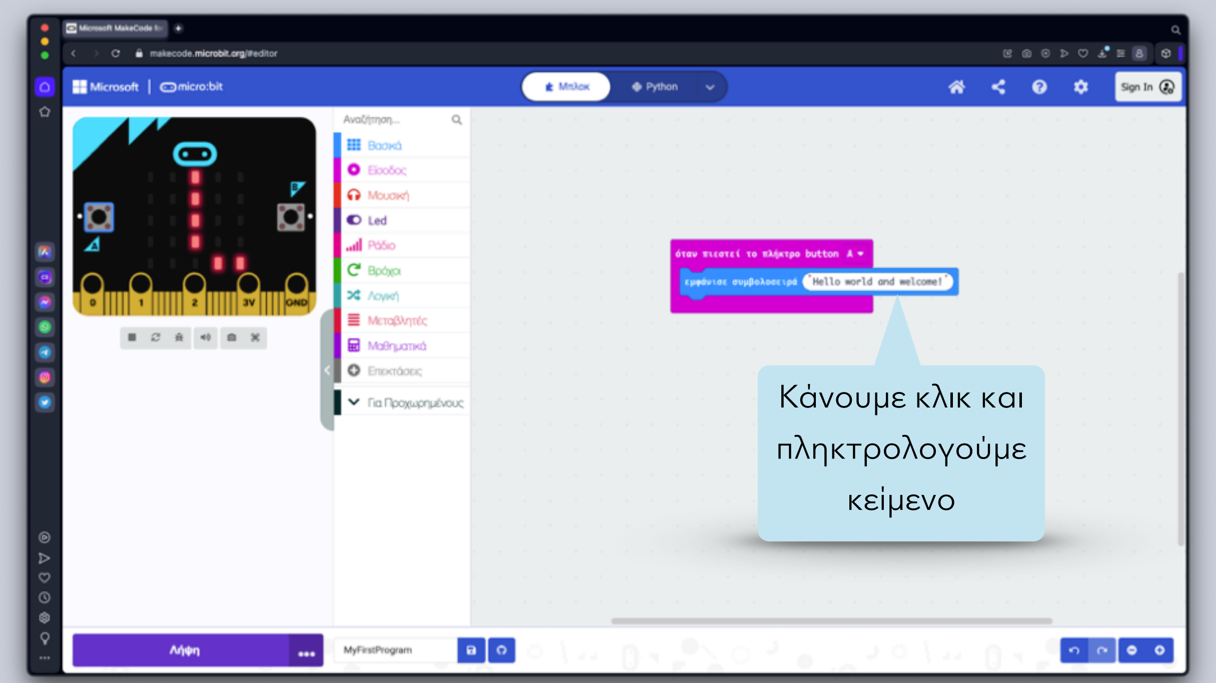
Από την κατηγορία "Βασικά", επιλέγουμε "εμφάνισε αριθμό" και το τραβάμε μέσα στο μπλοκ "όταν πιεστεί το πλήκτρο Α". Για να ελέγξουμε τον κώδικα μας, πατάμε το κουμπί Α στο εικονικό micro:bit.

Αν προσθέσουμε και άλλο μπλοκ εμφάνισης αριθμού, τότε ο επόμενος αριθμός θα εμφανιστεί -σωστά το μαντέψατε!- μετά από τον πρώτο.

Η οθόνη LED του micro:bit είναι πολύ μικρή, μόλις 5Χ5. Έτσι, ένας μεγάλος αριθμός δεν μπορεί να εμφανιστεί ολόκληρος σε αυτά τα LED. Αυτό που συμβαίνει είναι να γίνεται "κύληση" (scrolling) των υπόλοιπων ψηφίων.

Με παρόμοιο τρόπο μπορούμε να εμφανίσουμε και κείμενο - φέρνουμε το μπλοκ "εμφάνισε συμβολοσειρά" μέσα στο μπλοκ Εισόδου, και πατάμε το κουμπί Α για να δούμε το αποτέλεσμα (εικόνα κάτω).





LED, Εικονίδια & Σχήματα

Το MakeCode περιλαμβάνει μια σειρά από έτοιμα εικονίδια (σχήματα) που εμφανίζονται στα LED του micro:bit.

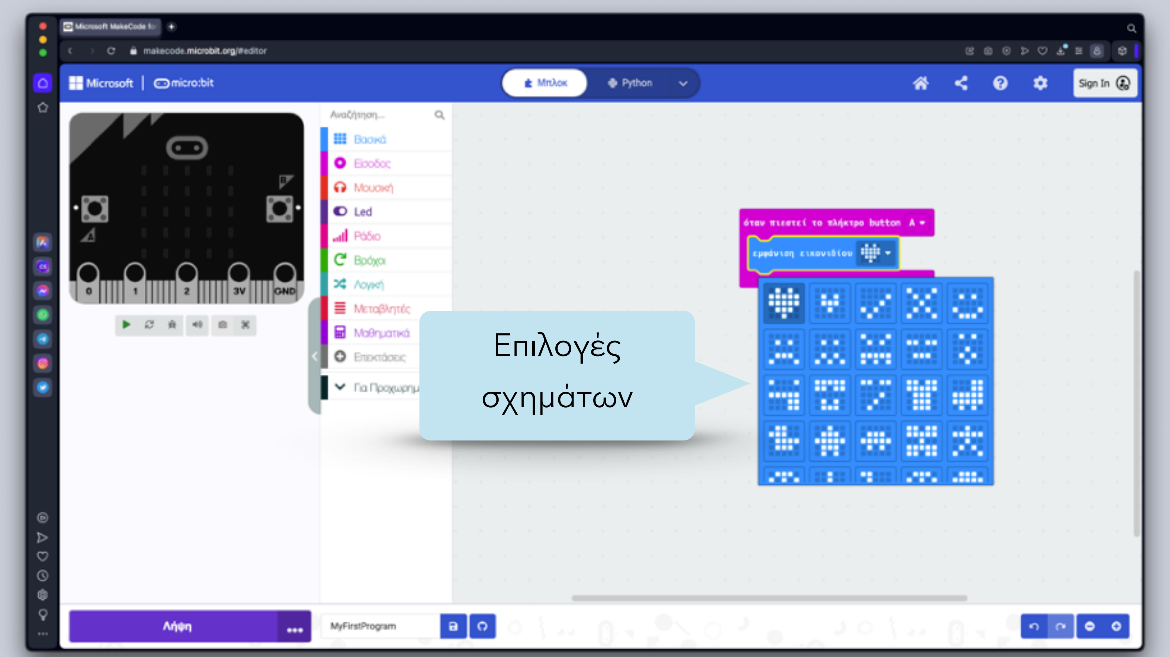
Μεταφέρουμε το μπλοκ "εμφάνιση εικονιδίου" στον χώρο εργασίας και μέσα στο μπλοκ Εισόδου "όταν πιεστεί το πλήκτρο A". Επιλέγουμε ένα από τα εικονίδια, ώστε με το πάτημα του πλήκτρου A να εμφανίζεται στα LED του micro:bit.

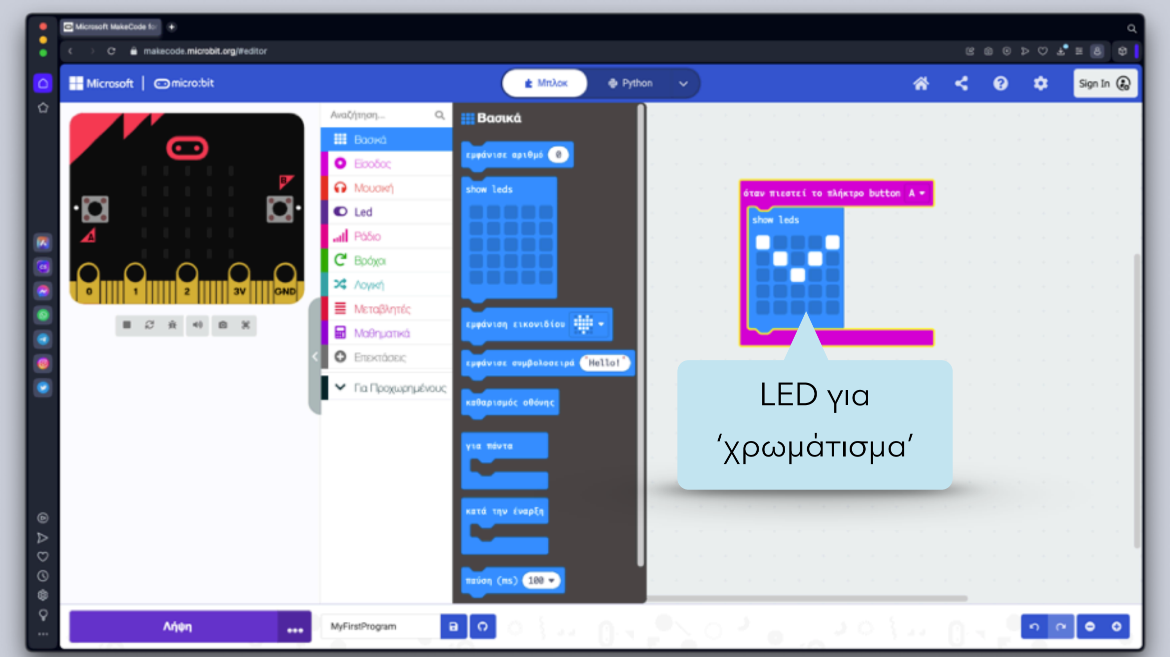
Αν μεταφέρουμε και άλλα μπλοκ, με διαφορετικά σχήματα, τότε το καθένα θα εμφανίζεται στα LED με τη σειρά που το έχουμε τοποθετήσει μέσα στο μπλοκ Εισόδου.

Με το μπλοκ "show leds" μπορούμε να δημιουργήσουμε (στο πλαίσιο 5X5) τα δικά μας σχήματα. Για κείμενο, χρησιμοποιούμε το μπλοκ "εμφάνισε συμβολοσειρά" της προηγούμενης σελίδας. Μεταφέρουμε το μπλοκ "show leds" στον χώρο εργασίας και μέσα στο μπλοκ Εισόδου.

Για να δημιουργήσουμε το σχήμα, κάνουμε κλικ σε κάθε τετραγωνάκι (LED) στο μπλοκ του χώρου εργασίας (εικόνα κάτω). Τα επιλεγμένα LED (αυτά που θα ανάβουν) εμφανίζονται με λευκό χρώμα. Αν θέλουμε να σβήσουμε κάποιο από αυτά, πατάμε ξανά πάνω του.

Και πάλι, μπορούμε να βάλουμε και δεύτερο μπλοκ (και περισσότερα), το ένα κάτω από το άλλο.





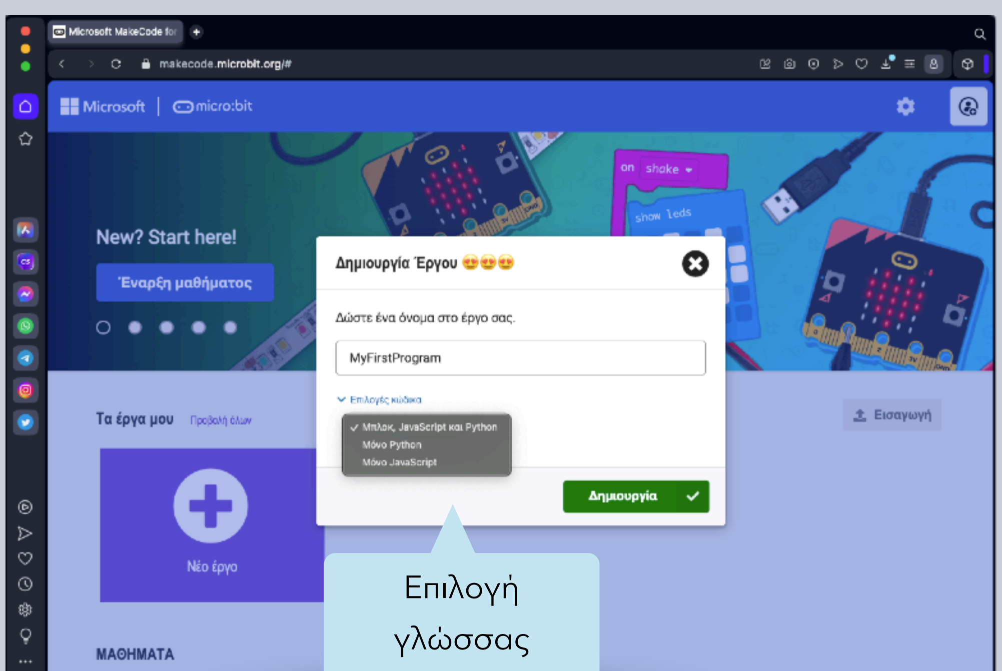
Από τα μπλοκ στην Python…

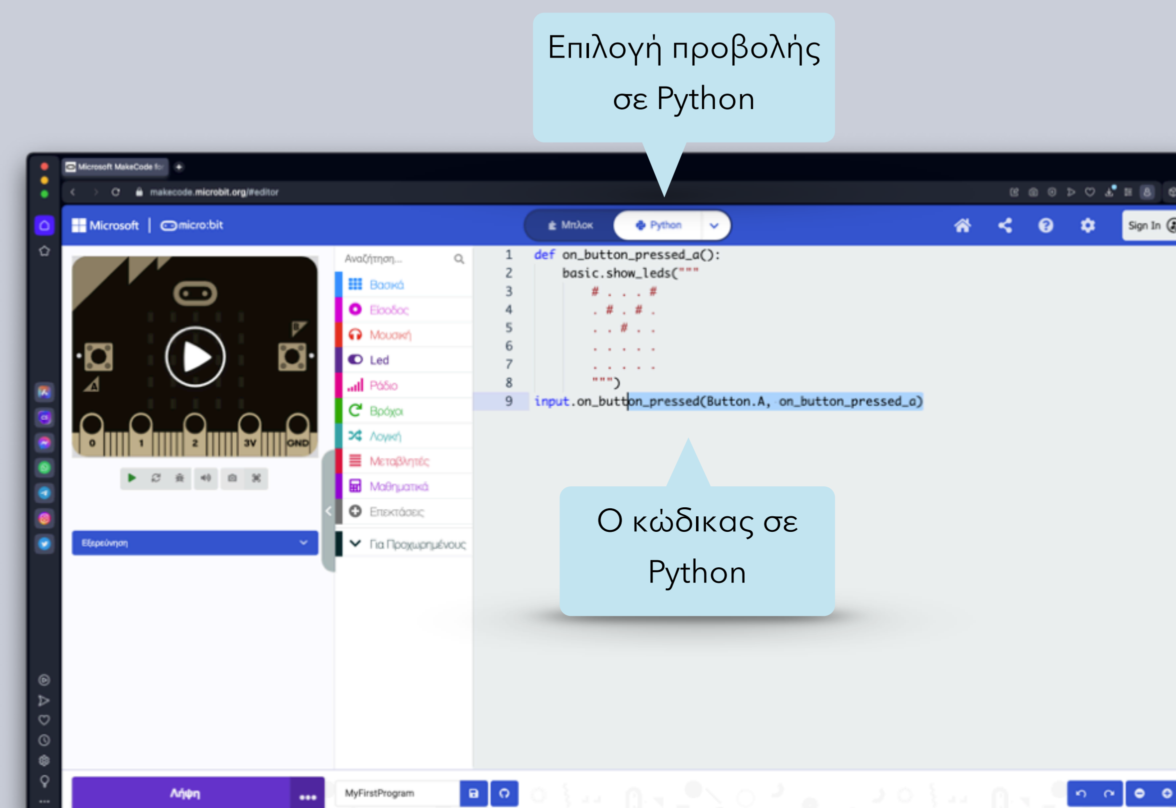
Με τη δημιουργία ενός νέου έργου (αρχική οθόνη MakeBlock) μπορούμε να επιλέξουμε κατά πόσο θα εμφανίζονται οι εντολές με Μπλοκ, Javascript και Python ή αν θα εμφανίζονται μόνο σε Javascript ή μόνο σε Python (εικόνα κάτω).

Η πρώτη επιλογή (μπλοκ & Javascript, Python) έχει ορισμένα σημαντικά πλεονεκτήματα: αν είμαστε αρχάριοι στον προγραμματισμό με Python, μπορούμε να βλέπουμε τις αντίστοιχες εντολές με μπλοκ, ώστε να ελέγχουμε τη σύνταξή μας. Επίσης, βλέπουμε τη σωστή σύνταξη συναρτήσεων (functions) που αφορούν μόνο το micro:bit.

Στο τελευταίο παράδειγμα της προηγούμενης σελίδας, χρησιμοποιήσαμε το μπλοκ "show leds" για να δημιουργήσουμε συγκεκριμένο σχήμα. Το σχήμα αυτό θα εμφανίζεται στα LED του micro:bit με το κουμπί A.

```
def on_button_pressed_a():
    basic.show_leds("""
        # . . . #
        . # . # .
        . . # . .
        . . . . .
        . . . . .
        """)
input.on_button_pressed(Button.A,
on_button_pressed_a)
```





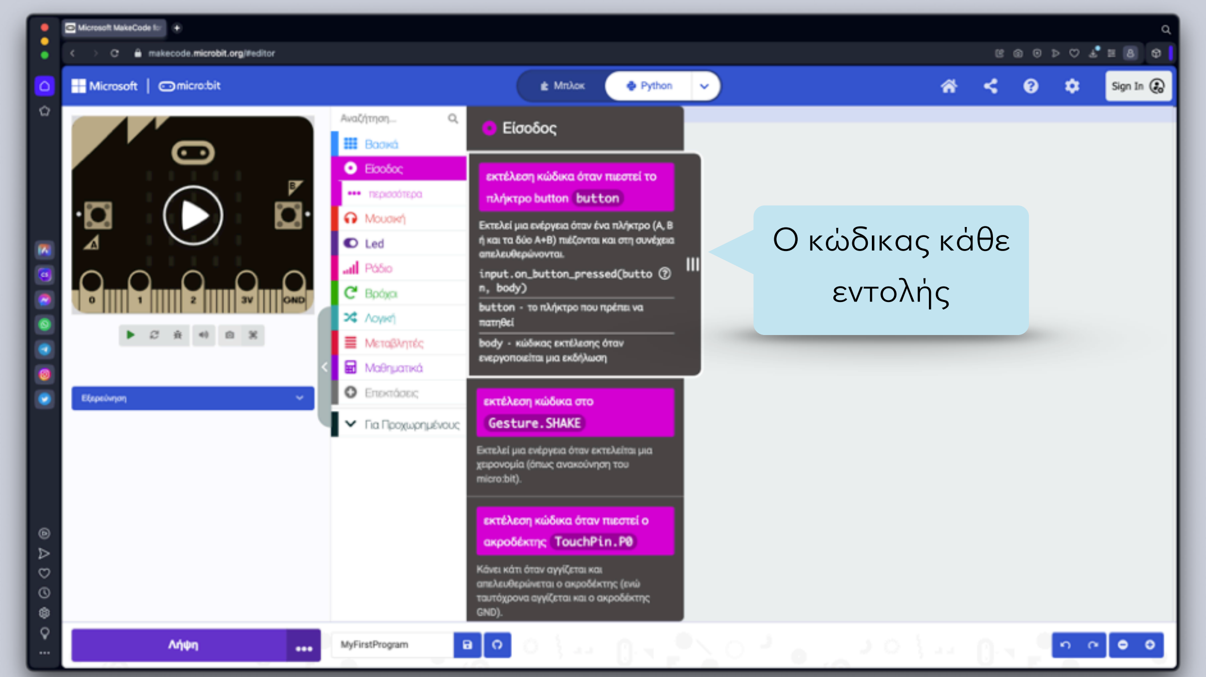
Κατηγορίες εντολών

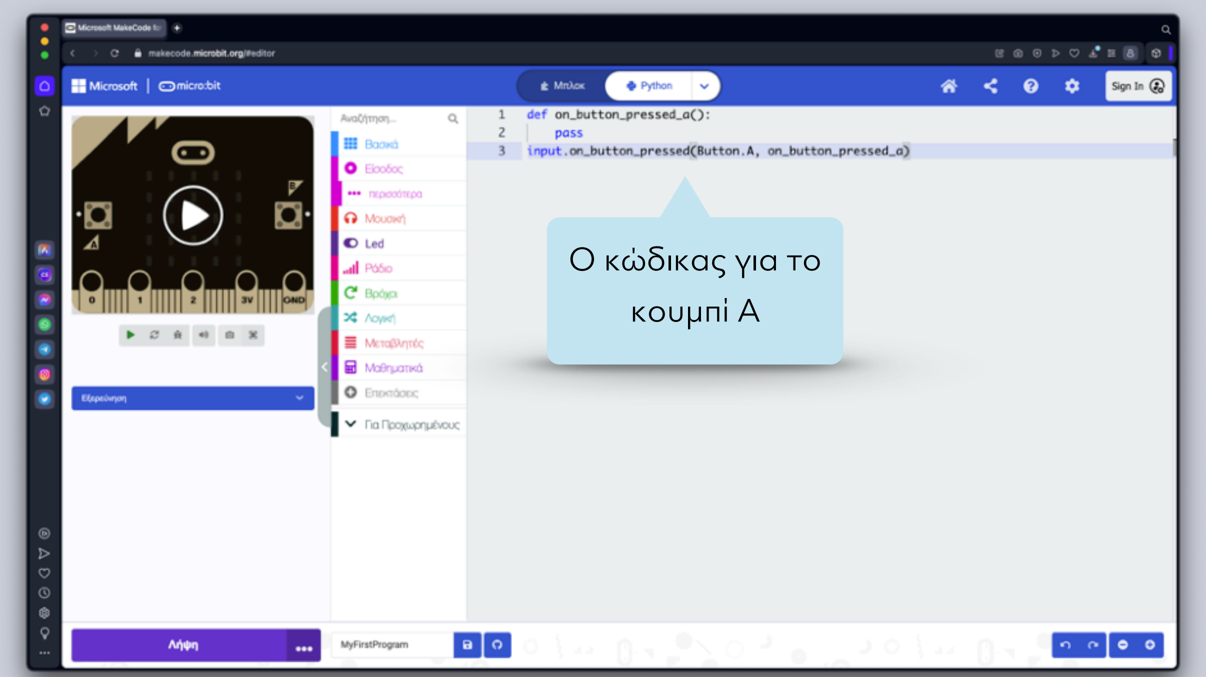
Το MakeBlock μας βοηθά να κατανοήσουμε και να χρησιμοποιήσουμε εύκολα (με απλό drag and drop) εντολές της Python. Όταν κάνουμε κλικ πάνω σε μια από τις κατηγορίες εντολών, αντί για μπλοκ (όπως είδαμε σε προηγούμενη σελίδα) εμφανίζονται οι εντολές σε Python. Αν κάνουμε κλικ σε μια από αυτές (εικόνα κάτω) εμφανίζονται περισσότερες πληροφορίες για τη συγκεκριμένη εντολή, καθώς και η σύνταξη του κώδικα της. Στον χώρο εργασίας μπορούμε, είτε να πληκτρολογήσουμε την εντολή, είτε να κάνουμε μεταφέρουμε τον κώδικα από την κατηγορία στην οποία είμαστε. Αν μεταφέρουμε τον κώδικα, αυτός εμφανίζεται στον χώρο εργασίας (εικόνα κάτω).

```
# Σχήμα που θα εμφανίζεται όταν πατήσουμε το κουμπί A

def on_button_pressed_a():
    pass
input.on_button_pressed(Button.A, 
on_button_pressed_a)
```

Στον κώδικα πιο πάνω έχουμε προσθέσει και σχόλιο.Τα σχόλια είναι απαραίτητα, ειδικά αν δημιουργήσουμε ένα σύνθετο πρόγραμμα, ώστε να κατανοούμε κάθε μέρος του προγράμματος μας τι δουλειά κάνει (και πώς!).





Τι μάθαμε μέχρι τώρα:

- Στο κεφάλαιο **"BBC micro:bit"** γνωρίσαμε τον μικρότερο υπολογιστή που σχεδιάστηκε ποτέ για εκπαιδευτικούς σκοπούς, και που χρησιμοποιείται σήμερα από εκατομμύρια παιδιά σε όλο τον κόσμο.
- Μελετήσαμε δύο περιβάλλοντα προγραμματισμού του micro:bit, το Python Editor και το Microsoft MakeCode. Και τα δύο επιτρέπουν τον προγραμματισμό σε Python, ενώ προσφέρουν πολλές ευκολίες για αρχάριους προγραμματιστές.
- Γνωρίσαμε τους αισθητήρες και τα άλλα μέρη του micro:bit και προγραμματίσαμε τα LED του να προβάλλουν εικόνες αλλά και κείμενο .
- Αξιοποιήσαμε τους αισθητήρες του micro:bit, όπως το επιταχυνσιόμετρο (accelerometer) για να δημιουργήσουμε ένα απλό παιχνίδι "πέτρα, ψαλίδι, χαρτί"



Δραστηριότητες

1. Να προγραμματίσετε το micro:bit, ώστε να αναβοσβήνουν συνεχώς τα LED του (για παράδειγμα η πρώτη και η τελευταία σειρά).

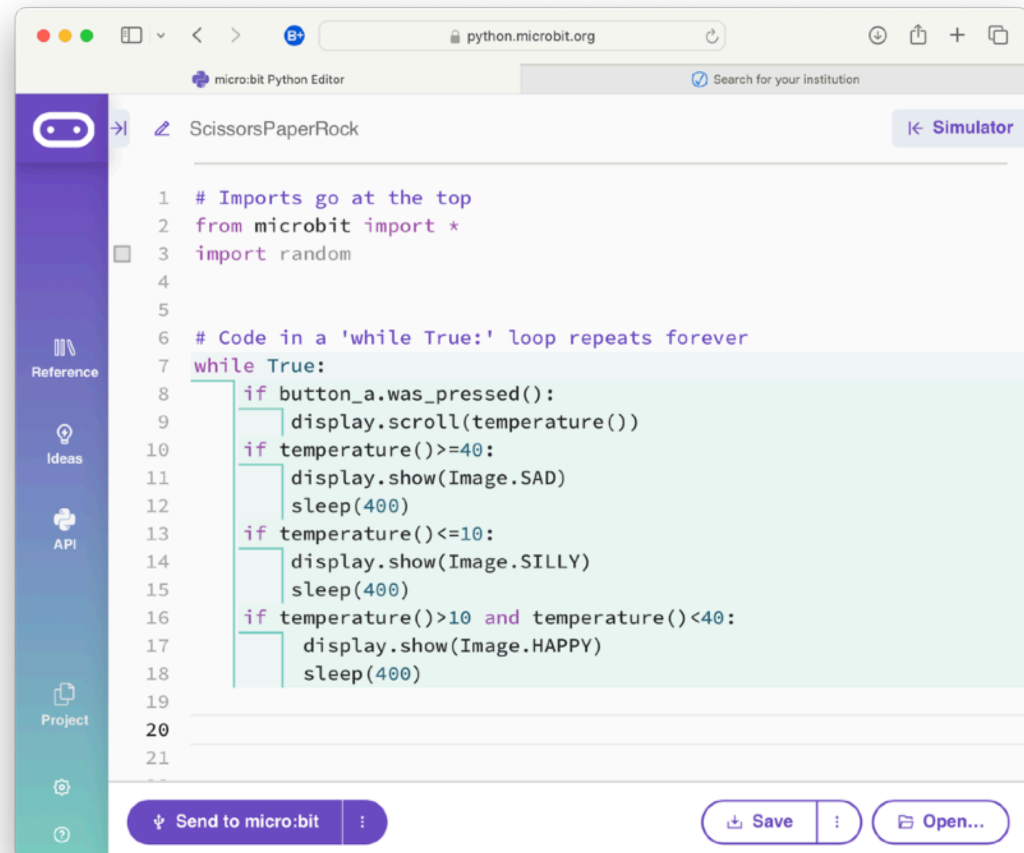
2. Να προγραμματίσετε το micro:bit ώστε να εμφανίζει μια εικόνα όταν πατάτε το κουμπί "A", άλλη εικόνα όταν πατάτε το "B" και άλλη εικόνα όταν πατάτε και τα δύο μαζί.

3. Να τροποποιήσετε το πρόγραμμα της πυξίδας, ώστε να δείχνει όλα τα σημεία του ορίζοντα (Βορρά, Νότο, Ανατολή, Δύση) και τα ενδιάμεσα σημεία (ΒΑ, ΒΔ, ΝΑ, ΝΔ).

4. Να χρησιμοποιήσετε την εντολή random για να δημιουργήσετε ένα ζάρι με το micro:bit. Δοκιμάστε να το χρησιμοποιήσετε σε παιχνίδια όπου χρειάζεται ζάρι (π.χ. Monopoly).

5. Να δημιουργήσετε ένα πρόγραμμα το οποίο να ελέγχει τη θερμοκρασία του δωματίου και να την εμφανίζει σε βαθμούς κελσίου στα LED του micro:bit.

6. Να εξηγήσετε τη λειτουργία του πιο κάτω προγράμματος, χωρίς να το εκτελέσετε:



MeetEdison

```
10 |  print ("Would you tell me where to go from here?")
20 |  print ("That depends on where you want to get to!")
30 | print  ("I don't care much where")
40 | print  ("Then it doesn't matter which way you go")
50 | #Alice in Wonderland
```

Meet Edison!

Η εκπαιδευτική ρομποτική είναι κάτι που αρέσει σε εκατομμύρια παιδιά σε όλο τον πλανήτη. Τα πιο συνηθισμένα ρομπότ που χρησιμοποιούμε είναι τα LEGO (EV3, WeDo 2.0, Spike Prime), τα Engino (στην Κύπρο), διάφορα μοντέλα βασισμένα σε Arduino (π.χ. mBot), ακόμη και ρομπότ τα οποία βασίζονται στο micro:bit. Υπάρχουν επίσης και τα BeeBot για μικρές ηλικίες, αλλά και άλλα πολλά.

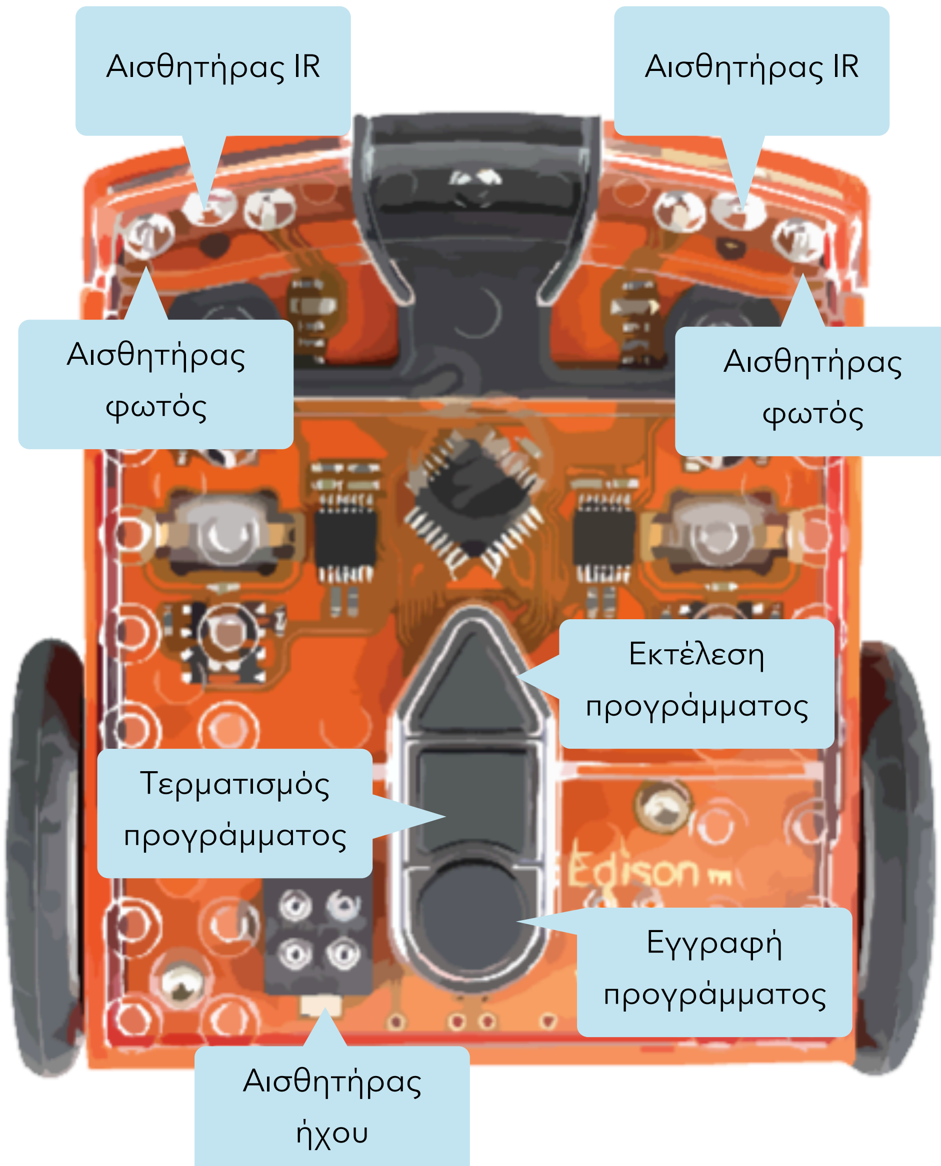
Ένα από αυτά τα ρομπότ είναι και το Edison, που μας έρχεται από την Αυστραλία. Το Edison είναι ένα πολύ μικρό (συγκριτικά) σε μέγεθος ρομπότ, χαμηλού κόστους, το οποίο μπορούμε να προγραμματίσουμε εύκολα, αλλά και να συνδυάσουμε με τουβλάκια LEGO για να δημιουργήσουμε μεγαλύτερες και πιο πολύπλοκες κατασκευές.

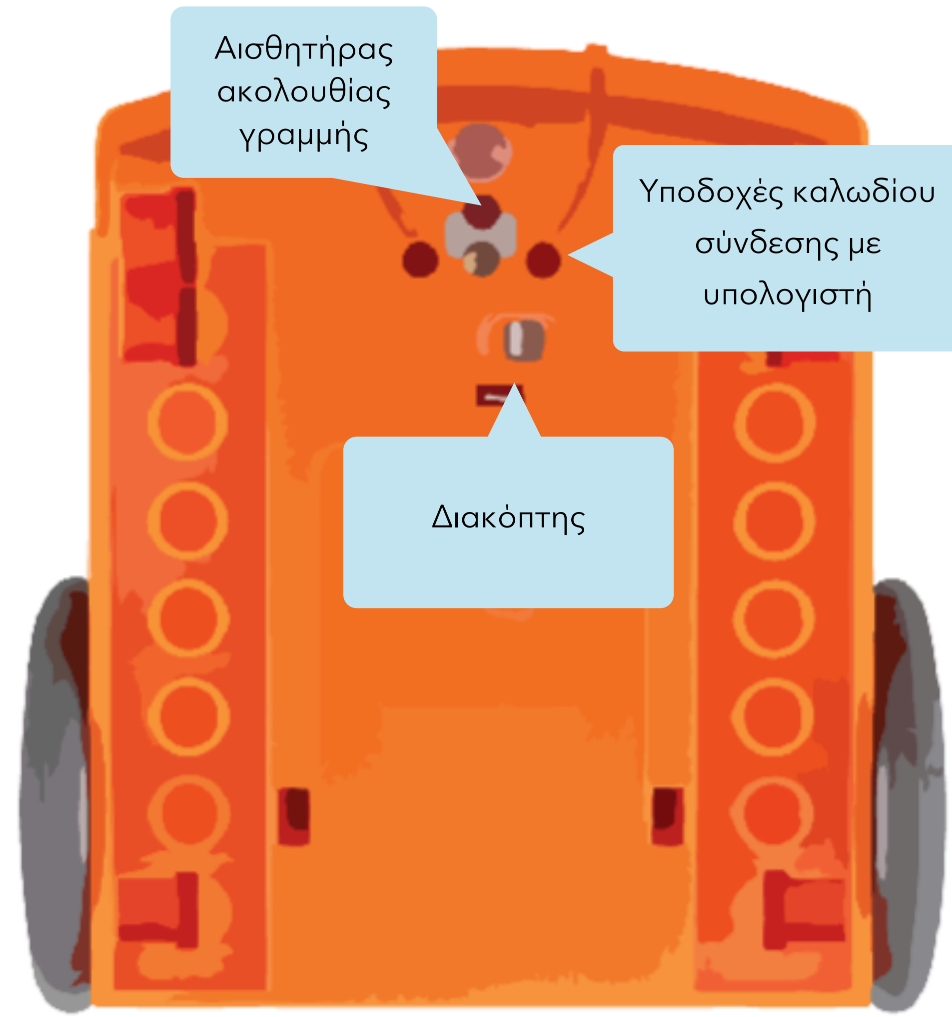


Τι θα γνωρίσουμε:

- Στο κεφάλαιο "**MeetEdison**" θα γνωρίσουμε το εκπαιδευτικό ρομπότ Edison, το περιβάλλον προγραμματισμού EdPy (Python), καθώς και τους αισθητήρες που διαθέτει.
- Θα προγραμματίσουμε το Edison με εντολές της Python ώστε να κινείται στο πάτωμα σε όλες τις κατευθύνσεις και να χρησιμοποιεί τα LED του.
- Θα χρησιμοποιήσουμε τις συναρτήσεις του Edison για να το προγραμματίσουμε να ακολουθεί μια μαύρη γραμμή, να αποφεύγει εμπόδια όταν κινείται και να ελέγχεται με το κτύπημα των χεριών μας.







Γνωριμία με το Edison

Το Edison, αν και μικρό σε μέγεθος και με χαμηλό κόστος, περιλαμβάνει αρκετούς αισθητήρες. Στο πάνω μέρος, υπάρχει αισθητήρας ήχου, ενώ στο μπροστινό πάνω μέρος, αριστερά και δεξιά, υπάρχουν αισθητήρες φωτός. Δίπλα από τους αισθητήρες φωτός υπάρχουν αισθητήρες που χρησιμεύουν τόσο για την ανίχνευση εμποδίων, όσο και για την αποστολή και παραλαβή σημάτων από άλλα Edison! Οι αισθητήρες αυτοί έχουν τη δυνατότητα να χρησιμοποιηθούν ακόμη και για έλεγχο του Edison μέσω του χειριστηρίου της τηλεόρασης!

Στο κάτω μέρος του Edison υπάρχει αισθητήρας που επιτρέπει να αναγνωρίζει γραμμή και να την ακολουθεί (εικόνα πάνω δεξιά).

Edison & Εκπαίδευση

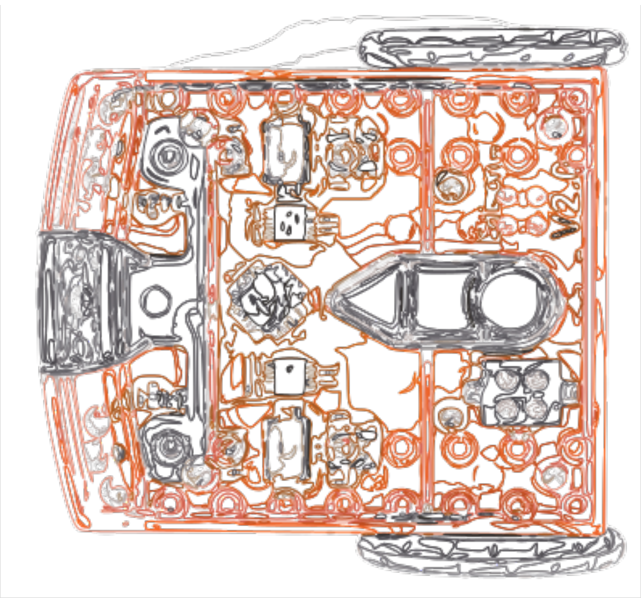
Συνήθως τα εκπαιδευτικά ρομπότ είναι αρκετά ακριβά ή αρκετά πολύπλοκα στον προγραμματισμό. Το Edison έχει σχεδιαστεί από την αρχή ώστε να είναι χαμηλού κόστους (κάτω από 50 ευρώ) και να μπορεί να προγραμματιστεί ακόμη και χωρίς υπολογιστή!

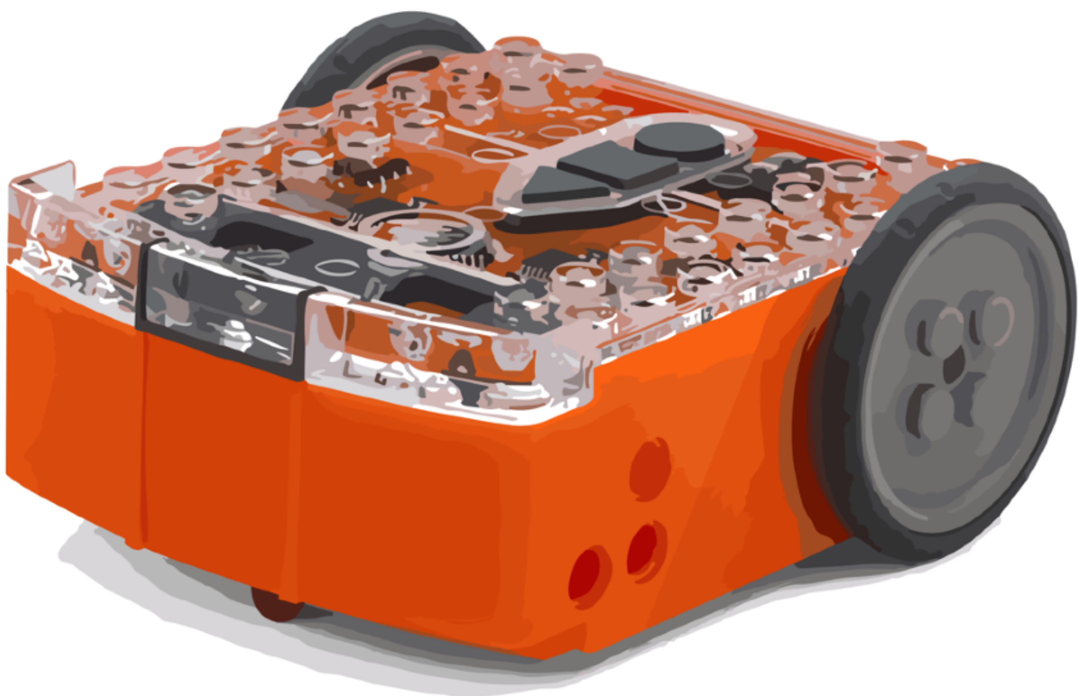
Μπορεί να “διαβάσει” ραβδωτό κώδικα (εικόνα δεξιά) ώστε να “μάθει” νέα προγράμματα (για παράδειγμα, πώς να αποφεύγει εμπόδια). Είναι τόσο απλό στη χρήση του, που μπορεί εύκολα να χρησιμοποιηθεί ακόμη και από παιδιά 4+ ετών!

Ακόμη ένα πλεονέκτημα του, είναι η δυνατότητα να το βελτιώσουμε με χρήση LEGO! Στο πάνω μέρος, αλλά και στα πλευρά, υπάρχουν υποδοχές για σύνδεση με LEGO. Αυτό επιτρέπει να κατασκευάσουμε πολύπλοκους μηχανισμούς με το Edison, φτάνει να έχουμε τα τουβλάκια.

Για προγραμματισμό, το Edison χρησιμοποιεί καλώδιο μεταφοράς ήχου! Όσο παράξενο και αν ακούγεται, οι εντολές μεταφέρονται ως ήχος στο ρομπότ μας!

Το Edison χρησιμοποιείται σήμερα σε αρκετά σχολεία σε πολλά μέρη του κόσμου. Στην Αυστραλία, στη χώρα που δημιουργήθηκε, το χρησιμοποιούν χιλιάδες παιδιά για να μάθουν προγραμματισμό και εκπαιδευτική ρομποτική. Η ευκολία προγραμματισμού του, και το χαμηλό του κόστος, βοηθούν στη χρήση του από παιδιά ακόμη και μικρότερης ηλικίας (4+).

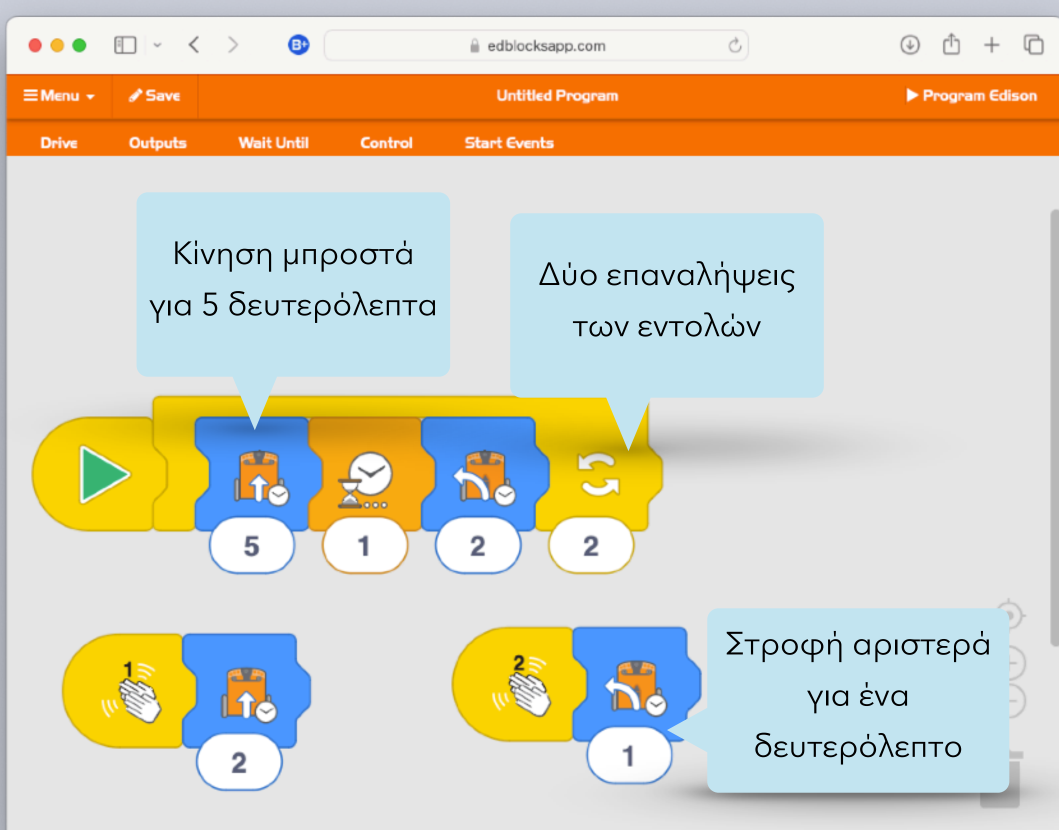




Edison & EdBlocks

Ο προγραμματισμός του Edison γίνεται με 4 διαφορετικούς τρόπους: για μικρότερες ηλικίες (προδημοτική, πρώτες τάξεις του δημοτικού) μπορούν να χρησιμοποιηθούν φύλλα εργασίας με ραβδωτό κώδικα (bar codes). Κάθε μια σειρά από bar codes αντιστοιχεί και σε ένα διαφορετικό πρόγραμμα (π.χ. έλεγχος του Edison με κτύπημα χεριών, όπως ο κώδικας της προηγούμενης σελίδας).

Ένα πολύ απλό περιβάλλον προγραμματισμού για το Edison είναι το EdBlocks (https://www.edblocksapp.com)



Το EdBlocks είναι πάρα πολύ απλό, καθώς δε χρησιμοποιεί καν κείμενο για περιγραφή των οδηγιών. Βασίζεται σε πολύ απλά σχήματα, έτσι είναι κατάλληλο ακόμη και για παιδιά της προδημοτικής.

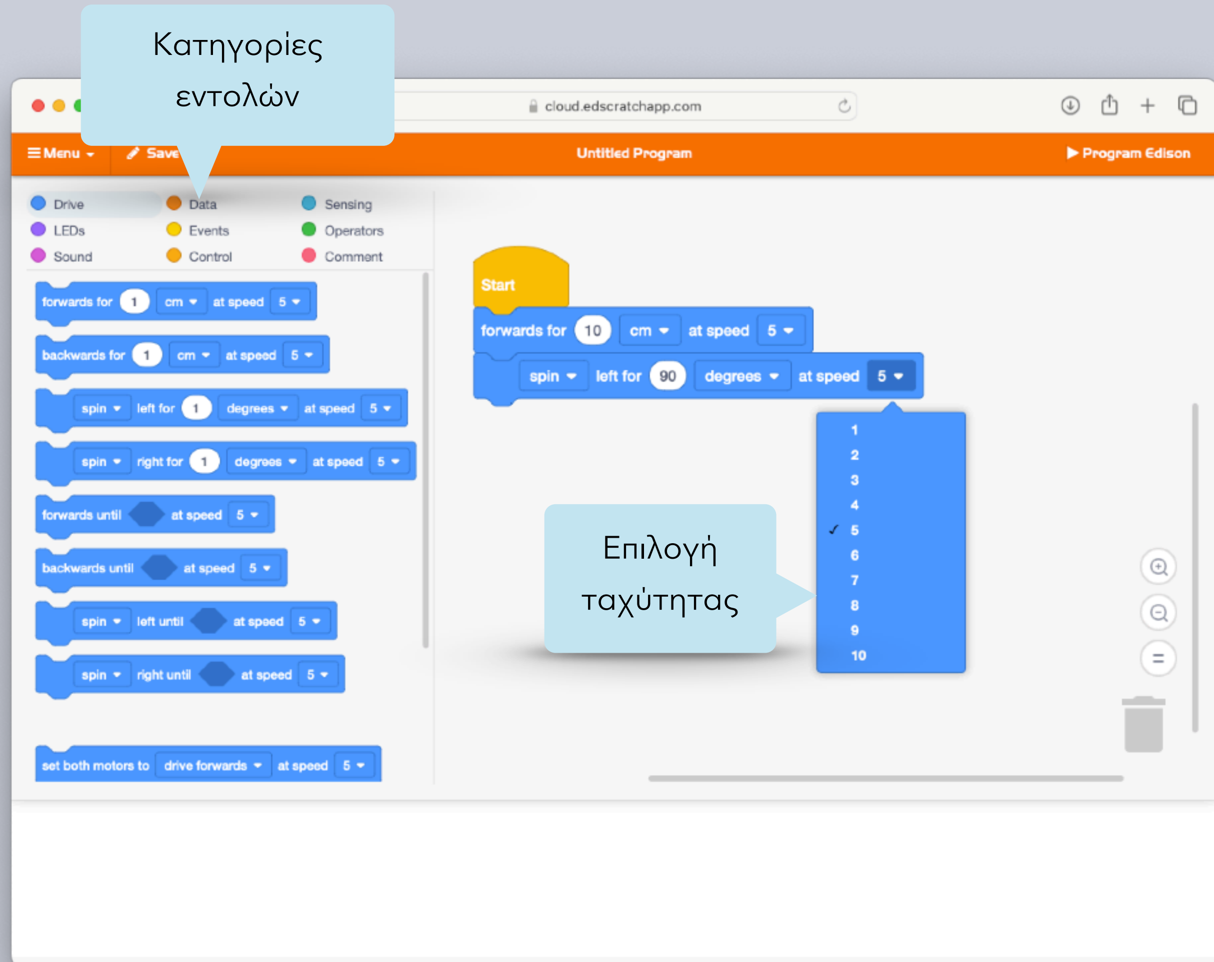
Οι εντολές είναι χωρισμένες σε 5 κατηγορίες. Κάθε εντολή έχει τη μορφή ενός μπλοκ. Μεταφέρουμε τα μπλοκ στο κύριο μέρος του παραθύρου, και τα ενώνουμε. Μεταφέρουμε τον κώδικα μέσω του καλωδίου ήχου και το εκτελούμε. Η κίνηση, ακόμη και οι στροφές, υπολογίζονται σε δευτερόλεπτα!



Edison & EdScratch

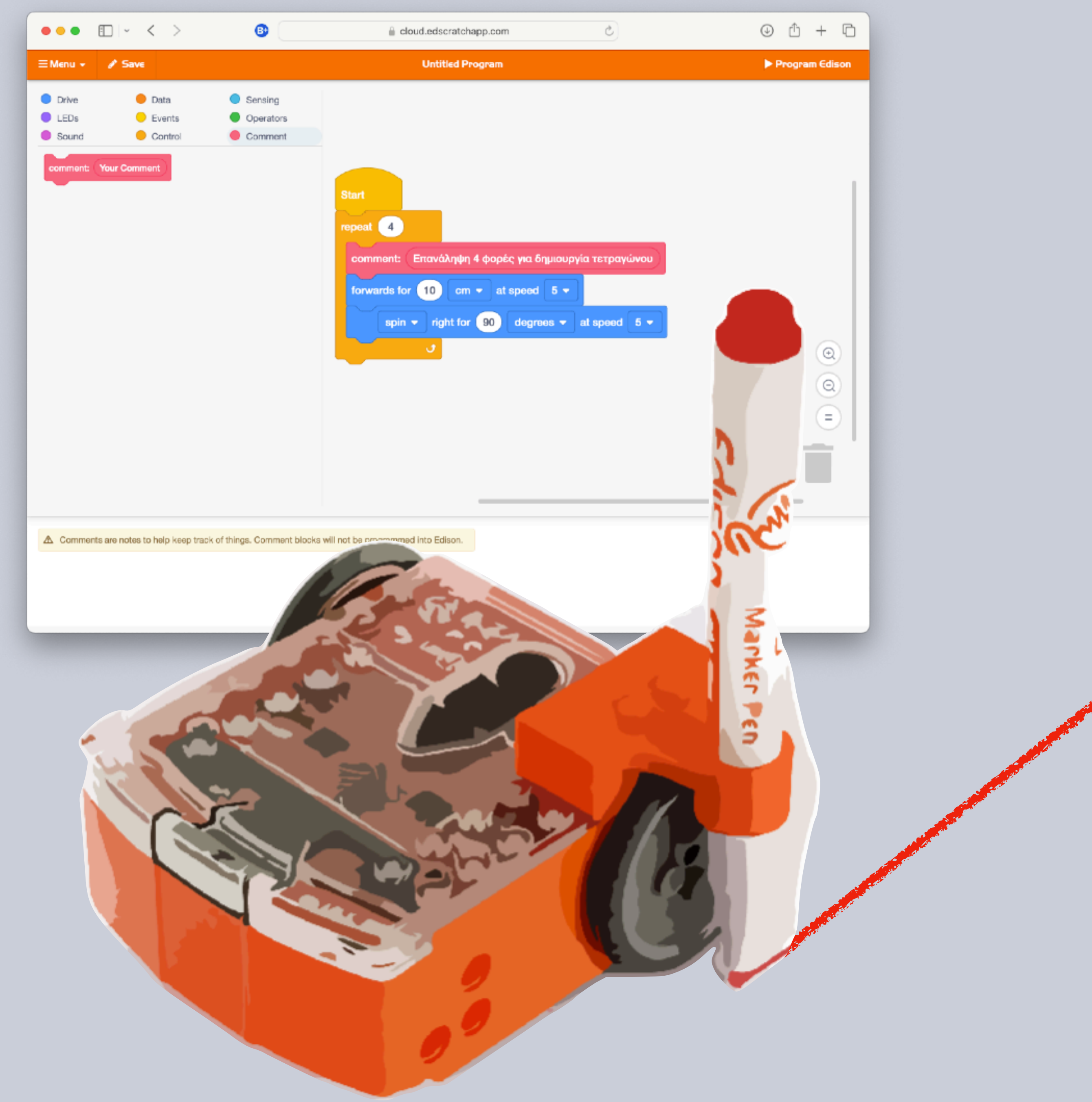
Το περιβάλλον του EdScratch βασίζεται -σωστά μαντέψατε- στο Scratch 3.0! Αν έχετε εμπειρία προγραμματισμού με τον γατούλη και την παρέα του Scratch, τότε θα βρείτε πολύ εύκολο τον προγραμματισμό του Edison με αυτό το περιβάλλον.

Ο προγραμματισμός γίνεται μέσω της διαδικτυακής εφαρμογής του EdScratch στη διεύθυνση: https://cloud.edscratchapp.com/



Οι δυνατότητες του EdScratch είναι πολύ μεγαλύτερες από του EdBlock. Μια από τις σημαντικότερες αλλαγές είναι να προγραμματίσουμε το Edison να κινείται με εκατοστόμετρα, αντί για δευτερόλεπτα.

Αυτό μας επιτρέπει να χρησιμοποιήσουμε το ρομπότ μας ακόμη και για τη δημιουργία σχημάτων πάνω σε χαρτί (κώδικας στην εικόνα πιο κάτω)!

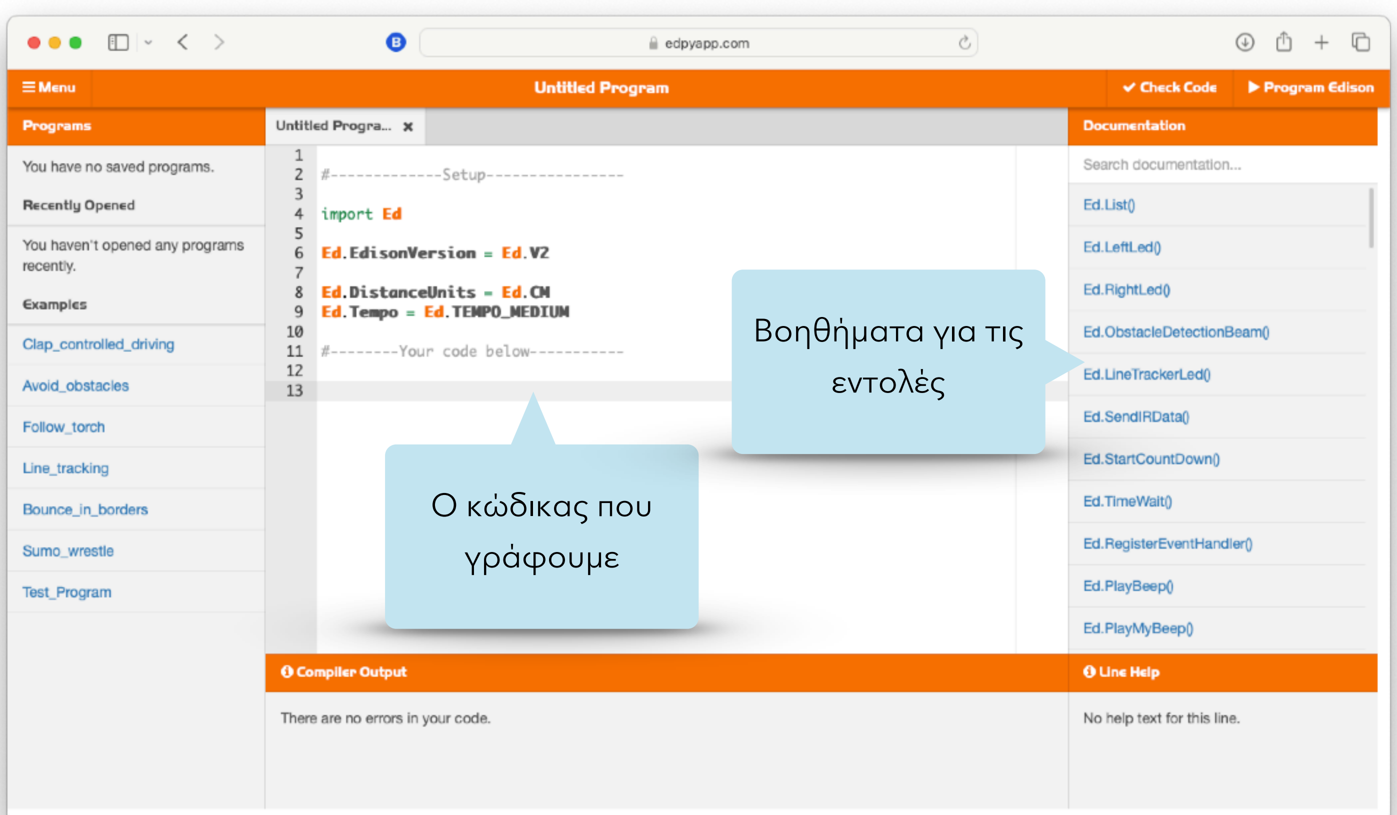


Edison & EdPy

Τα μικρά πορτοκαλί ρομπότ είναι η μεγάλη αγάπη κάθε πύθωνα! Λατρεύουν να παίζουν μαζί τους. Για την ακρίβεια, κάθε καλός πύθωνας έχει ένα μικρό πορτοκαλί ρομπότ στην παρέα του. Και ο καλύτερος τρόπος να παίξουν -και να επικοινωνήσουν μαζί του- είναι μέσω του EdPy!

Το EdPy είναι περιβάλλον προγραμματισμού για το Edison, με χρήση εντολών Python (γι'αυτό και το αγαπούν οι πύθωνες τόσο πολύ). Τρέχει μόνο μέσω της σελίδας:

https://edpyapp.com/





Το EdPy, αν και σίγουρα πιο δύσκολο στη χρήση από τα προηγούμενα (EdBlocks, EdScratch) προσφέρει περισσότερες δυνατότητες. Μπορούμε να αλλάξουμε τα πάντα, όλες τις εντολές, με πολύ μεγάλη ακρίβεια.

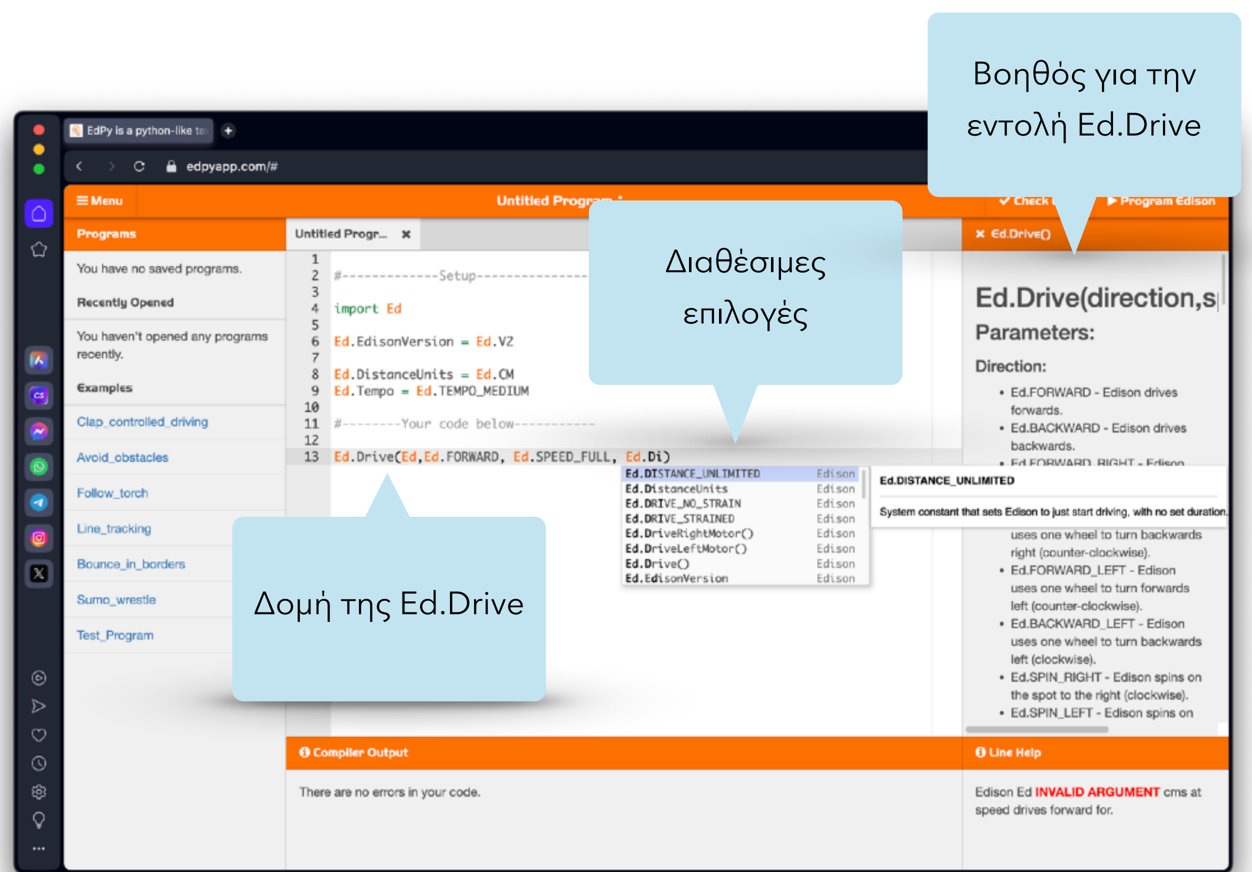
Το EdBlocks είναι το πιο απλό περιβάλλον προγραμματισμού του Edison. Ακολουθεί το EdScratch, που είναι πιο πολύπλοκο, αλλά με περισσότερες δυνατότητες. Τέλος, το EdPy, που βασίζεται στην Python, είναι το πιο δύσκολο στη χρήση αλλά με πολύ μεγάλες δυνατότητες. Το EdPy περιλαμβάνει πολύ καλά και αναλυτικά βοηθήματα για τις συναρτήσεις και εντολές, όπως βλέπουμε στην εικόνα κάτω αριστερά. Μοιάζουν με τα αντίστοιχα του micro:bit, αν και δεν μπορούμε να φέρουμε τις εντολές στο πρόγραμμα μας με drag and drop.



EdDrive: ας κινηθούμε!

Ως πρώτο πρόγραμμα, θα κινήσουμε το Edison μπροστά, με μέγιστη ταχύτητα και για απεριόριστη απόσταση (καλά, μη νομίζετε ότι θα πάει και πολύ μακριά, κάπου θα κολλήσει…).

Καθώς πληκτρολογούμε μια εντολή, εμφανίζονται αυτόματα όλες οι σχετικές. Αυτό διευκολύνει πολύ την εργασία μας, καθώς δε χρειάζεται να θυμόμαστε τη σύνταξη κάθε εντολής (αν και είναι καλό να γνωρίζουμε τουλάχιστο τις εντολές κίνησης).



Με την Ed.Drive, μπορούμε να επιλέξουμε αν το Edison θα κινηθεί σε ευθεία γραμμή μπροστά (Ed.FORWARD) ή πίσω (Ed.BACKWARD). Με παρόμοιο τρόπο, μπορούμε να κινηθούμε μπροστά και αριστερά (Ed.FORWARD_LEFT) ή δεξιά (Ed.FORWARD_RIGHT) ή προς τα πίσω (Ed.BACKWARD_LEFT και Ed.BACKWARD_RIGHT αντίστοιχα).

Η ταχύτητα είναι σημαντική επιλογή: Ed.SPEED_FULL σημαίνει ότι το Edison κινείται και με τα δύο μοτέρ σε πλήρη ταχύτητα. Εναλλακτικά, μπορούμε να αλλάξουμε την ταχύτητα από το 1 ως το 10 (π.χ. Ed.SPEED_2).

Τέλος, ορίζουμε την απόσταση (σε εκατοστόμετρα). Είτε πληκτρολογούμε τον αριθμό (των cm) είτε την εντολή Ed.DISTANCE_UNLIMITED. σε μια τέτοια περίπτωση, δε θα σταματήσει να κινείται στην κατεύθυνση που δώσαμε στην Ed.Drive.

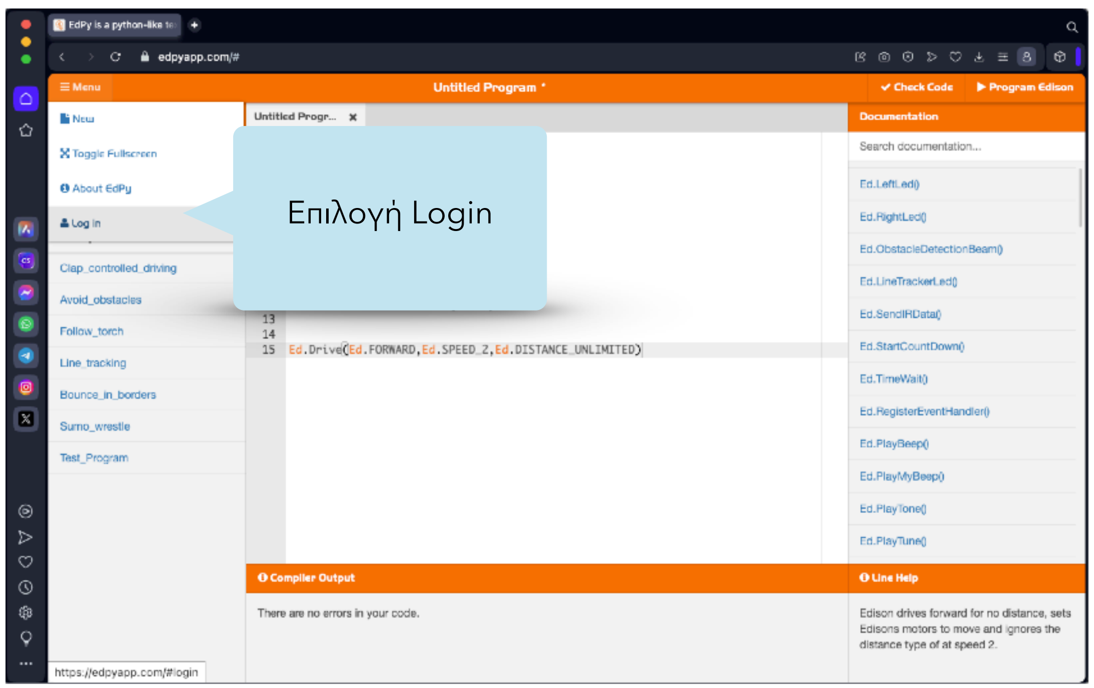
Η εντολή για την κίνηση (προς τα μπροστά) είναι:

```
Ed.Drive(Ed.FORWARD,Ed.SPEED_2, 5).
```

Με την πιο πάνω εντολή, το Edison θα κινηθεί μπροστά, με ταχύτητα 2 (από τα 10) και για 5 εκατοστόμετρα.

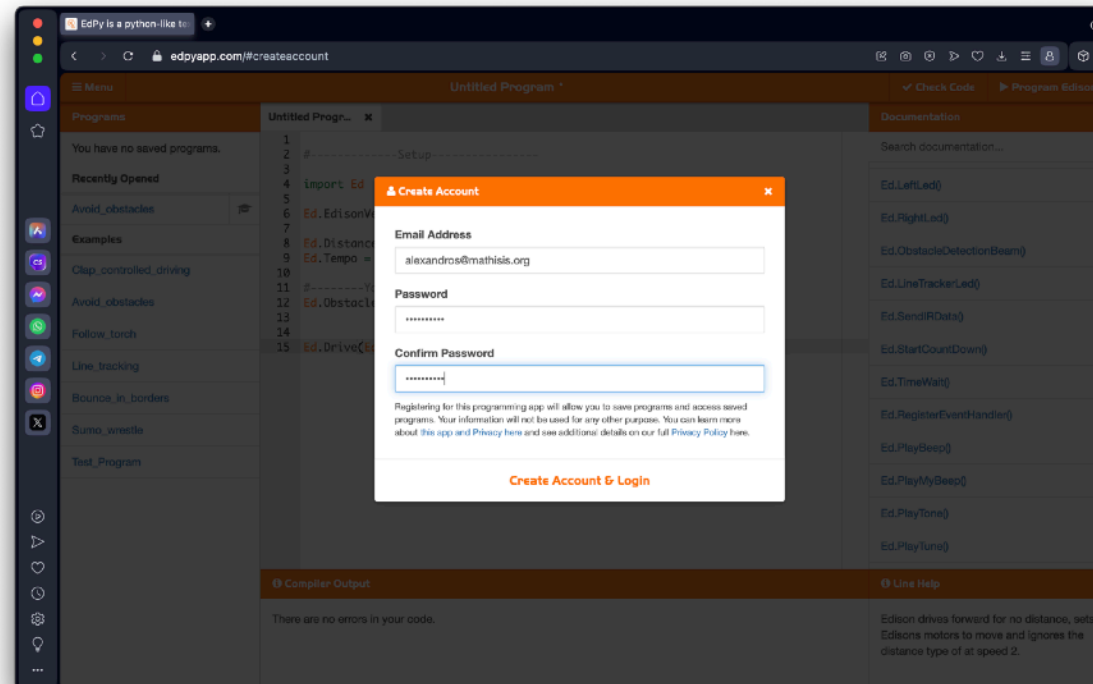
Αποθήκευση κώδικα

Στην προηγούμενη σελίδα, γράψαμε το πρώτο μας (απλό) πρόγραμμα για Edison. Πριν προχωρήσουμε, θα πρέπει να αποθηκεύσουμε το πρόγραμμα μας. Για να γίνει αυτό, θα πρέπει να δημιουργηθεί λογαριασμός στη σελίδα του EdPy (εικόνα πιο κάτω).

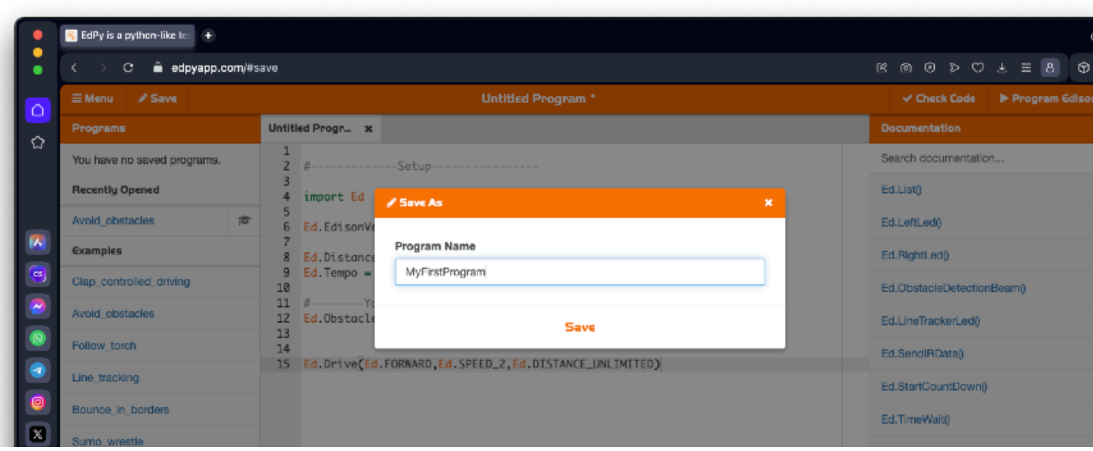


Από το "Menu" επιλέγουμε Log in. Αν έχουμε λογαριασμό, δίνουμε τα στοιχεία μας. Διαφορετικά, από το πλαίσιο που εμφανίζεται, επιλέγουμε "Create account". Η διαδικασία είναι απλή και διαρκεί λίγα δευτερόλεπτα (**σημείωση**: είναι καλό ενήλικας να δημιουργήσει τον λογαριασμό του παιδιού του).

Αφού δώσουμε τα στοιχεία μας, κάνουμε κλικ στην επιλογή "Create Account and Login" για δημιουργία του λογαριασμού και είσοδο σ'αυτόν.



Με την είσοδο στον λογαριασμό, εμφανίζεται η επιλογή "Save". Κάνουμε κλικ στο "Save" για να γίνει αποθήκευση του προγράμματός μας (εικόνα κάτω).



Αποφυγή εμποδίων

Το Edison διαθέτει αισθητήρες IR στο μπροστινό του μέρος (αριστερά και δεξιά), όπως είδαμε σε προηγούμενη σελίδα. Οι αισθητήρες αυτοί μπορούν να χρησιμοποιηθούν για πολλούς σκοπούς (π.χ. να στείλουν και να λάβουν μηνύματα από άλλα Edison, να ελέγξουμε το ρομπότ μας μέσω του χειριστηρίου της τηλεόρασης κ.α.). Ο κύριος σκοπός τους είναι η αναγνώριση και αποφυγή εμποδίων.

Πριν τη χρήση των αισθητήρων IR, θα πρέπει να τους ενεργοποιήσουμε. Αυτό γίνεται με την πιο κάτω εντολή:

```
Ed.ObstacleDetectionBeam(Ed.ON)
```

Με την εντολή αυτή ενεργοποιούνται οι αισθητήρες, και μας επιτρέπει να χρησιμοποιήσουμε εντολές που αφορούν ανίχνευση εμποδίων. Η ενεργοποίηση γίνεται στην αρχή του προγράμματος και πριν από τον βρόχο επανάληψης while True.

Το Edison μπορεί να χρησιμοποιήσει τους αισθητήρες IR για μόνο μια λειτουργία κάθε φορά. Δεν μπορεί, για παράδειγμα, να ελέγχει για εμπόδια και να στέλνει ή να λαμβάνει μηνύματα από άλλα Edison.

Αφού ενεργοποιήσουμε τους αισθητήρες, θα πρέπει να αρχίσει το Edison να ελέγχει για εμπόδια. Ο έλεγχος θα γίνεται συνέχεια, έτσι θα πρέπει να χρησιμοποιήσουμε ένα while True: βρόχο, για να επαναλαμβάνεται συνεχώς.

```
while True:
    Ed.Drive(Ed.FORWARD, Ed.SPEED_5,
    Ed.Distance_UNLIMITED)
```

Με τις πιο πάνω εντολές, το Edison κινείται μπροστά, με ταχύτητα 5, συνέχεια (Distance_UNLIMITED). Τώρα θα προσθέσουμε εντολή για ανίχνευση εμποδίων μπροστά από το ρομπότ μας:

```
if Ed.ReadObstacleDetection() > Ed.OBSTACLE_NONE:
```

Όταν δεν υπάρχει εμπόδιο μπροστά, η που ανιχνεύει την πορεία του Edison, ReadObstacleDetection() επιστρέφει Ed.OBSTACLE_NONE. Αν υπάρχει εμπόδιο μπροστά, τότε επιστρέφει Ed.OBSTACLE_AHEAD. Με την πιο πάνω συνθήκη ελέγχουμε κατά πόσο υπάρχει κάποιο εμπόδιο ή όχι, ώστε να δώσουμε εντολές (π.χ. να σταματήσει το ρομπότ) για να το αποφύγει.

```
Ed.Drive(Ed.STOP)
```

Οδήγηση με... παλαμάκια!

...έτσι μια μέρα αποφάσισαν οι πύθωνες να παίξουν ένα παιχνίδι: να ελέγχουν το Edison με το κτύπημα των χεριών τους (οι πύθωνες έχουν χέρια, αλλά αποφεύγουν τις χειραψίες και τα κρύβουν*).

Έτσι σκέφτηκαν να προγραμματίσουν το ρομπότ τους να κινείται μπροστά όταν χτυπούν δύο φορές τα χέρια τους, και να στρίβει αριστερά (άλλος είπε να στρίβει δεξιά) όταν χτυπούν μια φορά.

Ευτυχώς για τους πύθωνες (που τελικά δεν έχουν χέρια, παρά τις φήμες...) το Edison έχει καλό αισθητήρα ήχου στο πίσω αριστερά του μέρος και μπορεί να "ακούσει" τα κτυπήματα. Ua χρησιμοποιήσουεμ στον κώδικα μας την εντολή Ed.ReadClapSensor() μέσα σε έναν while True: βρόχο.



Με την Ed.ReadClapSensor() ελέγχουμε αν ακούγεται ήχος. Επειδή ο έλεγχος γίνεται συνέχεια, πρέπει να μπαίνει σε βρόχο επανάληψης. Όμως, είναι σημαντικό να υπάρχει και εκτός του βρόχου, ώστε να "καθαρίζει" από προηγούμενη χρήση.

***Όχι, οι πύθωνες δεν έχουν χέρια...**

Οι εντολές μας είναι οι ακόλουθες:

```
Ed.ReadClapSensor()

while True:
    Ed.ReadClapSensor()
```

Με την πιο πάνω εντολή, ελέγχει κατά πόσο έχει εντοπιστεί ήχος από τον αισθητήρα.

```
    if Ed.ReadClapSensor() == Ed.CLAP_DETECTED:
```

Δημιουργήσαμε μια συνθήκη σε περίπτωση που έχει ανιχνευθεί ήχος.

```
        Ed.Drive(Ed.FORWARD, Ed.SPEED_5, 10)
```

Σε περίπτωση που έχει ανιχνευθεί ήχος, τότε το Edison θα προχωρήσει μπροστά με ταχύτητα 5, για 10cm.

Με την εντολή Ed.TimeWait() μπορούμε να κάνουμε το Edison να περιμένει λίγο χρόνο (χιλιοστά δευτερολέπτου) ώστε να εντοπίσει και δεύτερο κτύπημα.

```
Ed.TimeWay(200,Ed.TIME_MILLISECONDS)
```



Το Edison στη γραμμή…

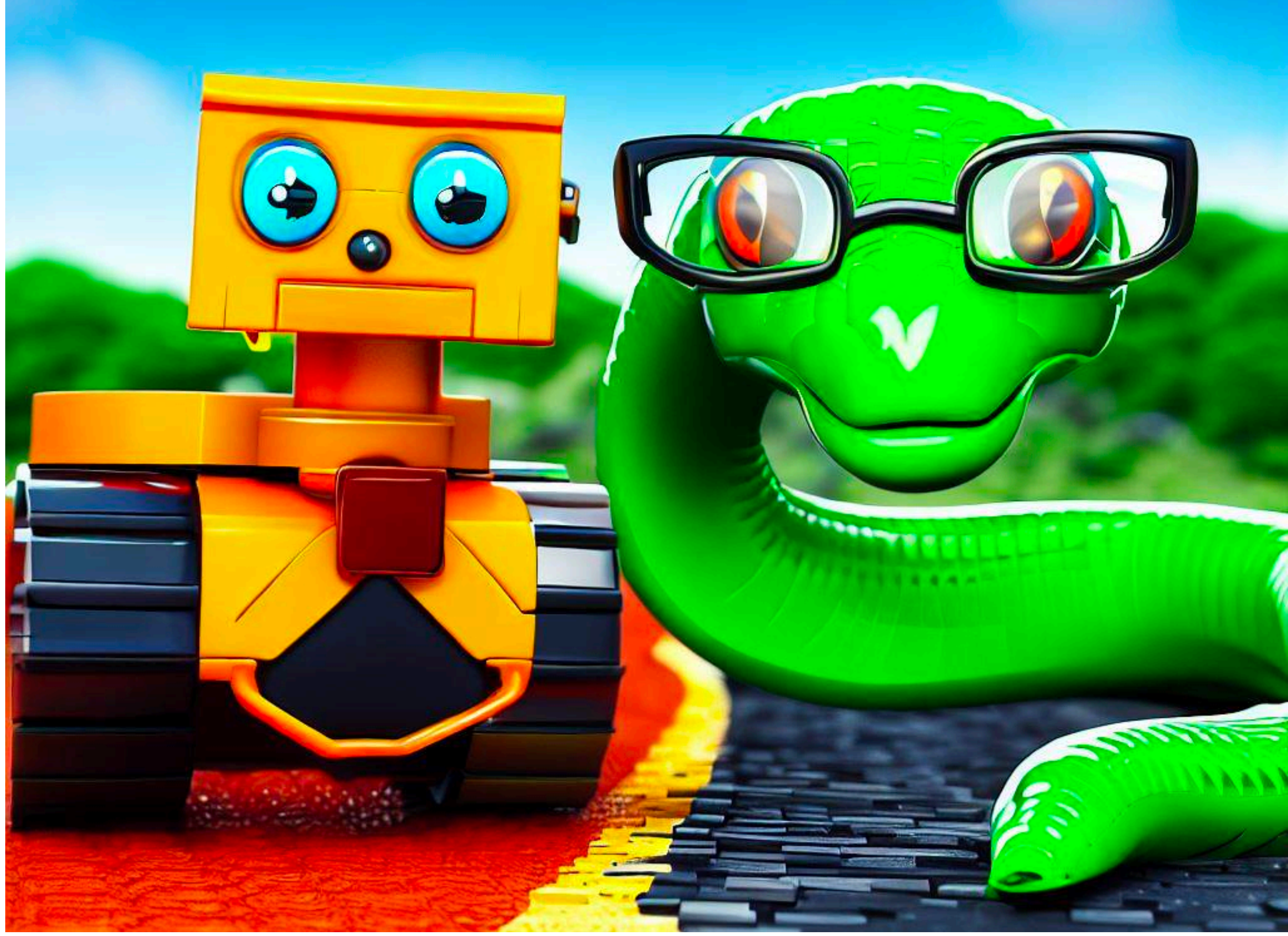
…από τότε που είδαν τον "Μάγο του Οζ", στους πύθωνες αρέσει να κινούνται πάνω σε κίτρινους δρόμους από τούβλα. Το Edison προτιμά τους δρόμους με μαύρο χρώμα. Προσπάθησαν να κινηθούν ο ένας στον δρόμο του άλλου, χωρίς αποτέλεσμα: το Edison δυσκολεύεται να "δει" οτιδήποτε εκτός από μαύρη γραμμή στο έδαφος!

Για να προγραμματίσουμε το Edison να κινηθεί και να ακολουθήσει μια μαύρη γραμμή, θα πρέπει να ενεργοποιήσουμε τον αισθητήρα που βρίσκεται στο κάτω μέρος του:

```
Ed.LineTrackerLed(Ed.ON)
```

Η εντολή αυτή θα πρέπει να βρίσκεται έξω από τον βρόχο επανάληψης. Όταν "βλέπει" λευκό χρώμα, θα πρέπει να προχωρά ευθεία και δεξιά. Αν δεν "βλέπει" λευκό χρώμα, τότε θα πρέπει να κινείται μπροστά και αριστερά. Με τον τρόπο αυτό, παραμένει πάντοτε στη μαύρη γραμμή:

```
while True:
    if Ed.ReadLineState() == Ed.LINE_ON_WHITE:
        Ed.Drive(Ed.FORWARD_RIGHT, Ed.SPEED_1,
        Ed.DISTANCE_UNLIMITED)
```



Η εντολή if ελέγχει αν ο αισθητήρας "βλέπει" λευκό χρώμα.Αν βρίσκεται σε λευκό χρώμα, τότε το Edison προχωρά μπροστά και δεξιά με ταχύτητα 1 (χαμηλή ταχύτητα). Μέχρι να βρει μαύρο χρώμα (Ed.LINE_ON_BLACK) θα συνεχίσει να κινείται μπροστά και δεξιά. Διαφορετικά (else:), το Edison θα προχωρά μπροστά και αριστερά. Έτσι, θα κινείται -με κάπως περίεργο τρόπο- πάνω στη μαύρη γραμμή.

```
    else:
        Ed.Drive(Ed.FORWARD_LEFT, Ed.SPEED_1,
        Ed.DISTANCE_UNLIMITED)
```

Τι μάθαμε μέχρι τώρα:

- Στο κεφάλαιο **"MeetEdison"** γνωρίσαμε ένα μικρό σε μέγεθος και χαμηλού κόστους, συμβατό με LEGO εκπαιδευτικό ρομπότ.
- Το Edison μπορεί να προγραμματιστεί με 4 διαφορετικούς τρόπους, συμπεριλαμβανομένης της Python, μέσα από το διαδικτυακό περιβάλλον EdPy.
- Το EdPy, όπως και το IDLE, είναι διαδραστικά περιβάλλοντα προγραμματισμού. Αυτό σημαίνει πως κάθε εντολή εκτελείται (και ελέγχεται για λάθη) καθώς την πληκτρολογούμε.
- Μέσα από το EdPy μπορούμε να προγραμματίσουμε την κίνηση του Edison, αλλά και τη συμπεριφορά του, μέσα από τη χρήση αισθητήρων IR, ακολουθίας γραμμής, ήχου και φωτός.





ΒΙΟΓΡΑΦΙΚΟ

Γεννήθηκα στη Λευκωσία, παραμονή Χριστουγέννων του 1974. Τον πρώτο μου υπολογιστή τον απέκτησα το 1988, ένα Atari 520STFM. Ξεκίνησα τον προγραμματισμό σχεδόν την ίδια χρονιά, με τη Metacomco BASIC και αργότερα με STOS BASIC. Ξεκίνησα σπουδές ως Ηλεκτρολόγος Μηχανικός στο Ανώτερο Τεχνολογικό Ινστιτούτο, όμως ξεκίνησα ξανά τις σπουδές μου στο Παιδαγωγικό Τμήμα του Πανεπιστημίου Κύπρου. Κατά τη διάρκεια των σπουδών μου απέκτησα τα πρώτα μου Macintosh και -με δικά μου δάνεια- την Hypercard 2.2 και αργότερα το Macromedia Director 3.0 studio. Συνέχισα σπουδές με μεταπτυχιακό στα Αναλυτικά Προγράμματα και Διδασκαλία (2007) και διδακτορικό στα Πληροφοριακά Συστήματα και Επικοινωνίες (2017).

Ως δάσκαλος, εργάζομαι από το 1999 σε σχολεία, με εξαίρεση το διάστημα 2009 - 2011 που ήμουν με απόσπαση στο Παιδαγωγικό Ινστιτούτο.

Στα ενδιαφέροντα μου είναι τα διαδικτυακά περιβάλλοντα μάθησης, καθώς και η ιστορία των υπολογιστών. Με τον παιδικό μου φίλο Νικόλα Κτενά, ιδρύσαμε το 2014 το πρώτο Μουσείο Υπολογιστών στην Κύπρο.

Αλέξανδρος Κοφτερός, PhD

alexandros@mathisis.org

https://www.facebook.com/alexandros.kofteros

https://mathisis.org

Η PYTHON ΜΕ ΑΠΛΑ ΛΟΓΙΑ

ΣΥΝΤΟΜΟΣ ΟΔΗΓΟΣ ΓΙΑ ΑΡΧΑΡΙΟΥΣ

Αλέξανδρος Κοφτερός, PhD

Το βιβλίο εισαγάγει μαθητές όλων σχεδόν των ηλικιών -range(10,100)- στον προγραμματισμό υπολογιστών και συσκευών με τη χρήση της Python. Η Python είναι μια ιδιαίτερη γλώσσα προγραμματισμού. Είναι μια γλώσσα υψηλού επιπέδου, πολύ κατανοητή ακόμη και από αρχάριους, παρόλα αυτά πολύ ισχυρή ώστε να δημιουργήσουμε σχεδόν τα πάντα.

ΜΕΣΑ ΑΠΟ ΤΟ ΒΙΒΛΙΟ:

- Γνωρίζουμε βασικές εντολές της Python
- Εργαζόμαστε με μεταβλητές και επαναλήψεις
- Δημιουργούμε τις δικές μας συναρτήσεις
- Προγραμματίζουμε συσκευές
- ...συμπαθούμε τους πύθωνες περισσότερο...

ISBN 978-9925-8055-0-1

